Titre: Title:	Mathematical Programming of Peirce-Smith Converting
Auteur: Author:	Alessandro Navarra
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Navarra, A. (2013). Mathematical Programming of Peirce-Smith Converting [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1230/

UNIVERSITÉ DE MONTRÉAL

MATHEMATICAL PROGRAMMING OF PEIRCE-SMITH CONVERTING

ALESSANDRO NAVARRA
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INDUSTRIEL)
AOÛT 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

MATHEMATICAL PROGRAMMING OF PEIRCE-SMITH CONVERTING

présentée par : NAVARRA Alessandro
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. ANJOS Miguel F., Ph.D., président
M. SAVARD Gilles, Ph.D., membre et directeur de recherche
M. AJERSCH Frank, Ph.D., membre et codirecteur
M. STUART Paul, Ph.D., membre
M. MUNZ Richard J., Ph.D., membre

In memory of our lost friend and mentor,
Ralph Harris...

ACKNOWLEDGEMENTS

In the spring of 2005, near the end of my undergraduate studies in materials engineering at McGill University, Professor Ralph Harris and researcher Ka Wing Ng introduced me to the problem of Peirce-Smith converter scheduling. This topic was prominently featured in my final undergraduate workterm report, submitted in the summer of 2005.

The following two years were devoted primarily to my Master's degree in applied mathematics at McGill University. Although my Master's research was not directly related to Peirce-Smith converting, I was briefly exposed to mathematical programming during the coursework. Ralph and Wing trusted my intuition that mathematical programming could be used to optimize converting schedules, and led me to Dr Joël Kapusta (Formerly of Air Liquide), who also had an interest in Peirce-Smith converting.

At *École Polytechnique*, I have been privileged by two outstanding (and patient) supervisors. Firstly, Frank Ajersch had been a longtime associate of Ralph, and a distinguished expert in Peirce-Smith converting. Frank agreed to have me as a student, and connected me to my other supervisor, Gilles Savard, an expert in mathematical programming. Ultimately, it was Gilles' description of the course offerings that convinced me to study in the *Département de mathématiques et de génie industriel*. Indeed, it has been an honour to study under such talented and passionate professors, including Guy Desaulniers, Dominique Orban, André Langevin, Charles Audet and Louis-Martin Rousseau.

In the spring of 2007, shortly before my doctoral studies, Ralph transferred from McGill University to the Royal Melbourne Institute of Technology. Even from a distance, Ralph was expected to have a leading role in my doctoral research. Yet he passed away suddenly on July 12$^{\text{th}}$, less than two months before the beginning of my doctoral studies.

Following the tragic passing of Ralph, it was unclear if the mathematical programming of Peirce-Smith converting would still be a viable line of research, especially since Wing had accepted a position at CANMET and was no longer available. However, Joël and Air Liquide agreed to support my work, in spite of the dramatic change in personnel.

Having now completed my doctoral studies, my sincere acknowledgement and gratitude are due to Ralph and Wing who introduced me to Peirce-Smith converting, as well as Joël, Frank and Gilles who supported me through difficult times. Financial support was provided by Air Liquide and the *Fonds québécois de la recherche sur la nature et les technologies*.

RÉSUMÉ

Le convertissage par méthode Pierce-Smith (PS) est l'étape clé de la production de cuivre et du nickel. Cette opération se poursuit par des étapes séquentielles et présentent un cas idéal pour la programmation mathématique. Ce travail démontre les complexités thermochimiques et les étapes du convertissage au moyen d'un programme linéaire (PL) en nombres entiers mixtes. Ceci est la première fois que le convertissage PS est abordé dans un cadre de programmation mathématique et représente un avancement majeur de l'application de la recherche opérationnelle aux étapes de production des fonderies de cuivre et de nickel.

Les résultats démontrent que le cadre mathématique est fonctionnel, et peut être utilisé quotidiennement pour la gestion optimale des séquences d'opération de l'élaboration de cuivre et de nickel. Le cadre est flexible quant à la définition des contraintes du système et de la fonction objective. Cette flexibilité évoque la formulation de divers modes d'opération des fonderies. Le cadre pourra être exploité en forme de logiciel industriel que les fonderies pourraient utiliser pour coordonner la production journalière, et de varier leur mode d'opération selon les conditions de l'usine et du marché.

Le cadre a été formulé suivant une méthodologie qui est typique de la programmation mathématique, mais qui n'avait jamais été adaptée au convertissage PS. Premièrement, le problème se pose en forme générale. En effet, Le problème de convertissage consiste de la coordination des convertisseurs PS avec d'autres opérations dans la fonderie afin de maximiser la production durant une période fixe, tout en respectant les contraintes chimiques, volumétriques et thermiques. Deuxièmement, les diverses composantes et dimensions du système sont représentées par des structures algébriques générales; c'est-à-dire, des ensembles, des paramètres et des variables. Troisièmement, ces composantes sont liées de telle manière à ce que la formulation puisse être supportée par des techniques de résolution.

Les techniques de résolution par la programmation linéaire (PL) en nombres entiers mixtes sont bien établies. Par contre, il a été nécessaire d'introduire des simplifications pour pouvoir résoudre le problème des convertissages par l'adoption d'un cadre hypothétique de PL en nombres entiers mixtes. En considérant la vaste gamme de problèmes qui ont été déjà abordées dans ce type de cadre, il semblait raisonnable que le convertissage PS nécessiterait seulement des simplifications mineures.

En fait, le nouveau cadre mathématique exige certaines simplifications. En particulier,

il impose une rigidité artificielle sur deux températures variables : la température nominale d'effluent gazeux, et la température d'écrémage de la scorie. Ces simplifications peuvent être considérées mineures, puisqu'il existe des logiciels qui ne traitent même pas ces quantités comme variables. En outre, il semble maintenant plausible de formuler un cadre non linéaire qui fournirait un traitement plus robuste et réaliste qui tient compte de ces températures.

Les résultats de cette étude ont un attrait considérable pour les industries de cuivre et de nickel, puisque l'opération efficace d'une fonderie dépend directement de la coordination quotidienne de la production. Suivant les succès du travail actuel, des efforts superficiels provoqueront des changements majeurs dans les opérations journalières des fonderies de cuivre et de nickel.

ABSTRACT

Peirce-Smith (PS) converting is central to the production of copper and nickel, and is a lucrative, yet previously undeveloped, context for mathematical programming. The thermochemical complexities of PS converting have now been represented within a mixed-integer linear program (MILP). This is the first time that PS converting has been treated within a mathematical programming framework, hence a major advancement in the operations research of copper and nickel smelters.

The MILP framework is now functional, and can be used to construct optimal daily production schedules. The framework offers flexibility in the definition of system constraints and objective functions. This flexibility can accommodate the formulation of alternative modes of operation for smelters. The MILP framework can now be marketed as industrial software, to produce optimal daily schedules, and allow smelters to change their mode of operation in accordance to plant and market conditions.

The framework has been created using a methodology that is typical of mathematical programming, but which had never been adapted to Peirce-Smith converting. Firstly, the problem has been posed in appropriately general terms; indeed, the PS Converter Problem is to coordinate Peirce-Smith converters with other objects in the smelter, so as to maximize production within a fixed period of time, while respecting chemical, volumetric and thermal constraints. Secondly, the various components and dimensions of the system have been represented using general algebraic structures, such as sets, parameters and variables. Thirdly, these components have been related to each other in a manner that can be supported by solution techniques.

The solution techniques for MILP are well established. However, it was initially unclear what degree of simplification would be required in order to fit the PS Converter Problem into a hypothetical MILP framework. Given the vast scope of problems that have already been treated using MILP, it seemed plausible that the PS Converter Problem would require only minor simplifications.

The new MILP framework has indeed required some simplification. More precisely, the framework imposes an artificial rigidity on two classes of temperature variables: nominal offgas temperatures, and skimming temperatures. These simplifications can be considered minor, since there are existing software tools that do not even treat these quantities as

variables. Moreover, it now seems plausible to extend the current formulation into a nonlinear framework that would provide a more intensive and realistic treatment of the offgas and skimming temperatures.

The results of this study have a considerable appeal to the copper and nickel industries, since the efficient operation of a smelter is directly linked to its scheduling practice. Following the successes of the current work, superficial efforts will cause major changes in the daily operations of copper and nickel smelters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Sets

$\mathcal{A}$	Assignments
$\mathcal{A}_\circ$	Assignments from the previous schedule that are considered in the current optimization
$\mathcal{A}_i$	Assignments of class i
$\mathcal{A}_{ij}$	Assignments of object (i, j)
$\mathcal{A}_{\text{PSC}}$	Assignments (transitions) of the PSC class
$\mathcal{C}$	Object classes
$\mathcal{D}_{ik}$	Dependency clauses for assignment type k of class i
$\mathcal{D}_{\text{PSC}k}$	Dependency clauses for transition type k of the PSC class
$\mathcal{E}$	Chemical elements
$\mathcal{E}_k$	Chemical elements constituting stream k
$\mathcal{F}$	Flow mechanisms
$\mathcal{F}_{\text{Blast}}$	Blast flow mechanisms
$\mathcal{F}_{\text{Ch}}$	Flow mechanisms acting on charge streams
$\mathcal{F}_{\text{DCh}}$	Flow mechanisms acting on discharge streams
$\mathcal{F}_{\text{Feed}}$	Flow mechanisms acting on feed strems
$\mathcal{F}_k$	Flow mechanisms acting on stream k
$\mathcal{F}_{\text{NGBlow}}$	Flow mechanisms acting on nongaseous streams that are fed during blowing actions
$\mathcal{F}_{\text{Offgas}}$	Offgas flow mechanisms
$\mathcal{F}_{\text{Prod}}$	Flow mechanisms acting on product streams
$\mathcal{F}^{\text{M}}$	Modulated flow mechanisms
$\mathcal{F}^{\text{M}}_{\text{Ch}}$	Modulated flow mechanisms acting on charge streams
$\mathcal{F}^{\text{M}}_{\text{DCh}}$	Modulated flow mechanisms acting on discharge streams
$\mathcal{F}^{\text{MSM}}$	Modulated and semi-modulated flow mechanisms
$\mathcal{F}^{\text{MSM}}_{\text{Ch}}$	Modulated and semi-modulated flow mechanisms acting on charge streams
$\mathcal{F}^{\text{MSM}}_{\text{DCh}}$	Modulated and semi-modulated flow mechanisms acting on discharge streams
$\mathcal{F}^{\text{SM}}$	Semi-modulated flow mechanisms
$\mathcal{F}^{\text{SM}}_{\text{Ch}}$	Semi-modulated flow mechanisms acting on charge streams

$\mathcal{F}^{\mathrm{SM}}_{\mathrm{DCh}}$	Semi-modulated flow mechanisms acting on discharge streams
$\mathcal{F}^{\mathrm{UM}}$	Unmodulated flow mechanisms
$\mathcal{F}^{\mathrm{UM}}_{\mathrm{Ch}}$	Unmodulated flow mechanisms acting on charge streams
$\mathcal{F}^{\mathrm{UM}}_{\mathrm{DCh}}$	Unmodulated flow mechanisms acting on discharge streams
$\mathcal{L}^{\mathrm{DTrans}}_{\mathrm{PSC}}$	Direct transition feasibility clauses implemented in general linear form
$\mathcal{L}^{\mathrm{Trans}}_{\mathrm{PSC}}$	Transition feasibility clauses implemented in general linear form
$\mathcal{R}$	Reaction regimes
$\mathcal{R}_j$	Reaction regimes in which species j is stable
$\mathcal{S}$	Species
$\mathcal{S}_{\mathrm{Feed}}$	Species that constitute the feed streams
$\mathcal{S}_k$	Species that constitute stream k
$\mathcal{S}_{\mathrm{NGProd}}$	Species that constitute the nongaseous product streams
$\mathcal{S}_{\mathrm{Prod}}$	Species that constitute the product streams
$\mathcal{S}_{\mathrm{RgProd}}$	Species that constitute the product streams, and whose stability depends on the reaction regime
$\mathcal{T}_i$	Assignment types of class i
$\mathcal{T}_{\mathrm{PSC}}$	PSC transition types
$\mathcal{T}_{\mathrm{PSC,Crit}}$	PSC transition types that constitute the critical stage (relevant to the SC-PSC problem)
$\mathcal{T}_{\mathrm{PSC,Empty}}$	PSC transition types that cause the object converter to be empty
$\mathcal{T}_{\mathrm{PSC,IDCh}}$	PSC transition types that include a discharging action, but do not result in an empty converter
$\mathcal{T}^{-}_{\mathrm{PSC}k}$	PSC transition types that may precede a transition of type k
$\mathcal{T}^{-}_{\mathrm{PSC,Crit}}$	PSC transition types that occur prior to the critical stage (relevant to the SC-PSC problem)
$\mathcal{T}^{+}_{\mathrm{PSC,Crit}}$	PSC transition types that occur after to the critical stage (relevant to the SC-PSC problem)
$\mathcal{Z}$	Streams
$\mathcal{Z}_{\mathrm{Blast}}$	Blast streams
$\mathcal{Z}_{\mathrm{Blow}}$	Streams that are red or removed during the blowing action
$\mathcal{Z}_{\mathrm{Ch}}$	Streams that are fed during the charging action
$\mathcal{Z}_{\mathrm{DCh}}$	Streams that are removed during the discharging action (is identical to $\mathcal{Z}_{\mathrm{Prod}}$)
$\mathcal{Z}_{\mathrm{Feed}}$	Feed streams
$\mathcal{Z}_{\mathrm{NG}}$	Nongaseous streams
$\mathcal{Z}_{\mathrm{NGFeed}}$	Nongaseous feed streams

$\mathcal{Z}_{\text{NGProd}}$	Nongaseous product streams
$\mathcal{Z}_{\text{Prod}}$	Product streams (is identical to $\mathcal{Z}_{\text{DCh}}$)

Mappings and Operators

$\text{srce}(j)$	Source stream of flow mechanism j
$\text{class}(l)$	Class of assignment l
$\text{obj}(l)$	Object of assignment l
$l-$	Predecessor of assignment l
$l+$	Successor of assignment l

Parameters

$\overline{d}$	Upper bound on the duration of assignments
$\underline{d}^k_{\text{Blow}}$, $\overline{d}^k_{\text{Blow}}$	Minimum and maximum duration of the blowing action, during a transition of type k
$\underline{d}^k_{\text{Ch}}$, $\overline{d}^k_{\text{Ch}}$	Minimum and maximum duration of the charging action, during a transition of type k
$\underline{d}^k_{\text{DCh}}$, $\overline{d}^k_{\text{DCh}}$	Minimum and maximum duration of the discharging action, during a transition of type k
$\underline{d}^k_i$, $\overline{d}^k_i$	Minimum and maximum duration of segment i, during a transition of type k
e_{O}	Oxygen efficiency
ϕ^k_j	Volume fraction of gaseous species j in the blast of a transition of type k
$\dot{h}^k_{\text{Blast}}$	Rate at which heat is introduced into the bath as part of the blast, during a the blowing action of a transition of type k
$\underline{h}^{\text{obj}(l)}$, $\overline{h}^{\text{obj}(l)}$	Lower and upper bound on the bath heat that is permitted in $\text{obj}(j)$
$\underline{h}^{\text{PSC}j}$, $\overline{h}^{\text{PSC}j}$	Lower and upper bound on the bath heat that is permitted in PSC j
$\overline{n}_{\text{Asgn}i}$	Upper bound on the number of assignments that a member of class i may undertake in the current schedule
$\overline{n}_{\text{Crit}}$	Maximum number of converters that may be simultaneously in the critical stage (relevant to the SC-PSC problem)
n_i	Number of members in object class i
$\dot{m}^k_{i\text{Blast}}$	Mass rate at which element i is introduced into the bath as part of the blast, during a the blowing action of a transition of type k

M_i	Atomic mass of element i
M_j	Molecular mass of species j
ρ_j	Density of species j
ρ_j^{Norm}	Density of gaseous species j, as measured under Normal conditions
ρ_k	Density of stream k
$\bar{t}$	Upper bound on the assignment completion times of the current schedule
t_{Begin}	Time at which the current schedule begins
t_{End}	Time at which the current schedule ends
$T^{\text{Blast}k}$	Blast temperature of transition type k
$\underline{T}^{\text{Blow}k}$, $\overline{T}^{\text{Blow}k}$	Minimum and maximum bath temperature that is permissible during the discharging action of a transition of type k
$T^{\text{DCh}k}$	Bath temperature at the end of the discharging action of a transition of type k (relevant when k is an intermediate discharge transition type, $k \in \mathcal{T}_{\text{PSC,IDCh}}$)
$\underline{T}^{\text{DCh}k}$, $\overline{T}^{\text{DCh}k}$	Minimum and maximum bath temperature that is permissible during the discharging action of a transition of type k
$\underline{T}^{\text{obj}(l)}$, $\overline{T}^{\text{obj}(l)}$	Lower and upper bound on the bath temperature that is permitted in obj(j)
$T^{\text{Offgas}k}$	Nominal offgas temperature of transition type k
$\underline{T}^{\text{PSC}j}$, $\overline{T}^{\text{PSC}j}$	Lower and upper bound on the bath temperature that is permitted in PSC j
$\underline{u}^{jk}$, $\overline{u}^{jk}$	Minimum and maximum number of units that can be delivered through flow mechanism j, during a transition of type k
$\underline{v}^{jk}$, $\overline{v}^{jk}$	Minimum and maximum volume that can be delivered through flow mechanism j, during a transition of type k
$\overline{v}^{\text{Blow}k,\text{PSC}j}$	Maximum bath volume that is permitted in PSC j during the blowing action of a transition of type k
$\overline{v}^{\text{Ch}k,\text{PSC}j}$	Maximum bath volume that is permitted in PSC j during the charging action of a transition of type k
v_u^j	Volume that is carried by a unit of a modulated flow mechanism j
$\overline{v}_u^j$	Maximum volume that can be carried by a unit of a semi-modulated flow mechanism j
$\dot{v}_{\text{Blast}}^{\text{Norm}k}$	Volumetric blast rate, during the blowing action of a transition of type k, measured under Normal conditions
$\overline{v}^{\text{obj}(l)}$	Maximum bath volume that is permitted in obj(j)

$\overline{v}^{\mathrm{PSC}j}$	Maximum bath volume that is permitted in PSC j
w_{ij}	Weight fraction of element i in species j
w_{ik}	Weight fraction of element i in stream k
w_{jk}	Weight fraction of species j in stream k
$w_{\mathrm{H}j}^{\mathrm{Blast}k}$	Specific heat content of blast species j, as measured at the blast temperature of transition type k
$\underline{w}_i^{\mathrm{Blow}k}$, $\overline{w}_i^{\mathrm{Blow}k}$	Minimum and maximum weight fraction of element i that is permissible in the bath of a converter that is completing the blowing action of a transition of type k
$\underline{w}_{\mathrm{H}j}^{\mathrm{Blow}k}$, $\overline{w}_{\mathrm{H}j}^{\mathrm{Blow}k}$	Minimum and maximum specific heat content of species j that is permissible during the blowing action of a transition of type k
$\underline{w}_{\mathrm{H}k}^{\mathrm{Blow}k'}$, $\overline{w}_{\mathrm{H}k}^{\mathrm{Blow}k'}$	Minimum and maximum specific heat content of stream k that is permissible during the blowing action of a transition of type k'
$\underline{w}_{ik}^{\mathrm{Blow}k'}$, $\overline{w}_{ik}^{\mathrm{Blow}k'}$	Minimum and maximum weight fraction of element i that is permissible in stream k of a converter that is completing the blowing action of a transition of type k'
$\underline{w}_{jk}^{\mathrm{Blow}k'}$, $\overline{w}_{jk}^{\mathrm{Blow}k'}$	Minimum and maximum weight fraction of species j that is permissible in stream k of a converter that is completing the blowing action of a transition of type k'
$\underline{w}_k^{\mathrm{Blow}k'}$, $\overline{w}_k^{\mathrm{Blow}k'}$	Minimum and maximum weight fraction of stream k that is permissible in the bath of a converter that is completing the blowing action of a transition of type k'
$w_{\mathrm{H}j}^{\mathrm{DCh}k}$	Specific heat content of species j during the discharging action of a transition of type k (relevant when k is an intermediate discharge transition type, $k \in \mathcal{T}_{\mathrm{PSC,IDCh}}$)
$\underline{w}_{\mathrm{H}j}^{\mathrm{DCh}k}$, $\overline{w}_{\mathrm{H}j}^{\mathrm{DCh}k}$	Minimum and maximum specific heat content of species j that is permissible during the discharging action of a transition of type k
$\underline{w}_{\mathrm{H}k}^{\mathrm{DCh}k'}$, $\overline{w}_{\mathrm{H}k}^{\mathrm{DCh}k'}$	Minimum and maximum specific heat content of stream k that is permissible during the discharging action of a transition of type k'
$\underline{w}_i^{\mathrm{DCh}k}$, $\overline{w}_i^{\mathrm{DCh}k}$	Minimum and maximum weight fraction of element i that is permissible in the bath of a converter that is completing the discharging action of a transition of type k
$\underline{w}_k^{\mathrm{DCh}k'}$, $\overline{w}_k^{\mathrm{DCh}k'}$	Minimum and maximum weight fraction of stream k that is permissible in the bath of a converter that is completing the discharging action of a transition of type k'

$\underline{w}_i^{\circ k},\ \overline{w}_i^{\circ k}$	Minimum and maximum weight fraction of element i that is permissible in the bath of a converter that is to begin a transition of type k
$\underline{w}_{ik}^{\circ k'},\ \overline{w}_{ik}^{\circ k'}$	Minimum and maximum weight fraction of element i that is permissible in stream k of a converter that is to begin a transition of type k'
$\underline{w}_{jk}^{\circ k'},\ \overline{w}_{jk}^{\circ k'}$	Minimum and maximum weight fraction of species j that is permissible in stream k of a converter that is to begin a transition of type k'
$\underline{w}_k^{\circ k'},\ \overline{w}_k^{\circ k'}$	Minimum and maximum weight fraction of stream k that is permissible in the bath of a converter that is to begin a transition of type k'
$w_{\mathrm{H}j}^{\mathrm{Offgas}k}$	Specific heat content of offgas species j, as measured at the nominal offgas temperature of transition type k
$w_{\mathrm{H}j}^{\mathrm{Ref}}$	Specific heat content of species j, as measured at the reference temperature, 298.15 K
$w_{\mathrm{H}k}^{\mathrm{Ref}}$	Specific heat content of stream k, as measured at the reference temperature, 298.15 K

Variables

$\beta_{\mathrm{Rg}k}^{l}$	Regime determinant that is 1 iff the bath of obj(l) is in reaction regime k at the end of transition l
$\beta_{\mathrm{Suppl'k'}}^{lk}$	Assignment support determinant that is 1 iff assignment l is of type k, and is supported by assignment l' that is of type k'
$\beta_{\mathrm{Type}k}^{l}$	Assignment type determinant that is 1 iff assignment l is of type k
d_{Crit}	Duration of the critical stage (relevant to the SC-PSC problem)
d_{Cycle}	Duration of the converting cycle (relevant to the SC-PSC problem)
d^{l}	Duration of assignment l
d_{Blow}^{l}	Duration of the blowing action of transition l (is equal to d_4^l)
d_{Ch}^{l}	Duration of the charging action of transition l (is equal to d_2^l)
d_{DCh}^{l}	Duration of the discharging action of transition l (is equal to d_6^l)
d_i^l	Duration of segment i of transition l
$d_{\circ}^{l}$	Duration of time between the end of transition $l-$ and transition l
h_{Blast}^{l}	Heat that is blown into the melt as part of the blast of transition l
h_{Ch}^{l}	Heat that is introduced into obj(l) as part of the charge streams of transition l
h_{DCh}^{l}	Heat that is removed from obj(l) as part of the discharge streams of transition l
$h_{\mathrm{Env}i}^{l}$	Heat that is lost from the bath of obj(l) to the environment during segment i of transition l

$h^l_{\text{Env}\circ}$ Heat that is lost from the bath of obj(l) to the environment in the time between the end of transition $l-$ and the beginning of transition l

h^l_{NGBlow} Heat that is introduced into obj(l) as part of the nongaseous blow streams of transition l

h^l_{Offgas} Heat that is convected out of obj(l) as part of the offgas stream of transition l

h^l_{Ret} Heat that is retained in the bath of obj(l) at the end transition l

$m^l_{i\text{Blast}}$ Mass of element i that is blown into the melt as part of the blast of transition l

$m^l_{i\text{Prod}}$ Mass of element i within the product streams of transition l

$m^l_{j\text{Prod}}$ Mass of species j within the product streams of transition l

$m^l_{j\text{RetProd}}$ Mass of species j within the retained product streams of obj(l) at the end of transition l

m^l_k Mass of feed stream k that participates in transition l

$m^l_{\text{Ret}k}$ Mass of feed stream k retained in the bath of obj(l) at the end of transition l

t^l Completion time of assignment l

Type^l Assignment type of assignment l

u^{jl} Number of delivery units used during transition l to carry srce(j) via mechanism j, to or from obj(l)

v^{jl} Volume of srce(j) delivered during transition l via mechanism j, to or from obj(l)

$w_i^{\text{Blow}l}$ Weight fraction of element i in the bath of obj(l) at the end of the blowing action of transition l

$w_{ik}^{\text{Blow}l}$ Weight fraction of element i in stream k of obj(l) at the end of the blowing action of transition l

$w_{jk}^{\text{Blow}l}$ Weight fraction of species j in stream k of obj(l) at the end of the blowing action of transition l

$w_k^{\text{Blow}l}$ Weight fraction of stream k in the bath of obj(l) at the end of the blowing action of transition l

$w_i^{\text{DCh}l}$ Weight fraction of element i in the bath of obj(l) at the end of the discharging action of transition l

$w_k^{\text{DCh}l}$ Weight fraction of stream k in the bath of obj(l) at the end of the discharging action of transition l

$w_i^{\circ l}$ Weight fraction of element i in the bath of obj(l) at the beginning of transition l

$w_{ik}^{\circ l}$	Weight fraction of element i in stream k of obj(l) at the beginning of transition l
$w_{jk}^{\circ l}$	Weight fraction of species j in stream k of obj(l) at the beginning of transition l
$w_{k}^{\circ l}$	Weight fraction of stream k in the bath of obj(l) at the beginning of transition l

LIST OF APPENDICES

CHAPTER 1

PEIRCE-SMITH CONVERTING AND EXTRACTIVE METALLURGY

1.1 Importance of PS Converting

1.1.1 Global Presence

Peirce-Smith (PS) converting is applied in roughly 75% of the world copper production and 50% of the world nickel production, within 25 countries and 6 continents [1, 2, 3]. For copper, this amounts to roughly 12.7 million tons per year, at 8 000 USD/ton [4]. For nickel, this amounts to roughly 1.05 million tons per year, at 17 600 USD/ton [5].

Peirce-Smith converting is formally an operational bottleneck [6]. It is preceded by smelting furnaces that act as a buffer; the furnaces can normally provide feed exceeding the immediate capacity of the converters. PS converting then sets the tempo for the downstream operations. Improvements in the PS converting thus translate into overall benefits to the entire smelter.

PS converting is widely accepted by an industry accustomed to its simplicity [7, 8, 9], and has not undergone any radical changes since its inception in 1909 (Appendix A.1). Instead, it has had a series of incremental ancillary equipment and operation strategies (Appendix A.2), which have been supported through modeling and simulation techniques (Appendix A.3).

There are several technologies competing with the PS converter, such as the Hoboken siphon converter, and various continuous converting technologies (Appendix A.5). Even if these alternatives are destined to replace PS converting, it would take decades to phase out existing PS installations. Incremental improvements of PS will therefore remain lucrative.

Technological stagnation is detrimental in such a competitive environment with the potential for incremental improvement. But this stagnation can be overcome with reliable and justifiable decision-making. Herein, software tools have become invaluable to investigate the avenues of process change [10, 11, 12].

1.1.2 Interdisciplinary Divide

In the year 2013, the pyrometallurgical community is not well-versed in mathematical programming. Likewise, industrial engineers, mathematicians and computer scientists do not

have the domain-specific knowledge that would lead them to model Peirce-Smith operations.

Pyrometallurgists are seemingly unaware of the type of problems, and the level of abstraction, that can be attained through mathematical programming. The conventional attempts to simulate and optimize Peirce-Smith operations have relied largely on matrix algebra, and the classical methods of calculus (e.g. Newton's Method, [13]).

These classical approaches have enhanced the scientific understanding of PS converting, but they are oblivious to sixty years of technical and computational advances in mathematical programming. For example, few metallurgists have ever heard of the Simplex Method (Appendix B.1).

Pyrometallurgists are adept at solving continuous mathematical problems, as they have a fairly robust background in differential and integral calculus. On the other hand, they have a limited understanding of discrete mathematical problems, as they are not usually trained in algorithm design. This gap in education has limited metallurgists with regard to "systems thinking", which is now accepted as a major underpinning of process design and operations management [14].

In very rough terms, calculus is to continuous solutions spaces, what algorithm design is to discrete solution spaces; the essential link between continuous and discrete problems is provided by mathematical programming. Thus mathematical programming can build upon the existing competencies of pyrometallurgists, expanding toward the design of algorithms.

PS converting is a context that blends continuous and discrete mathematics, hence demanding the use of mathematical programming. This context is particularly attractive to nonferrous pyrometallurgists, because of its historical importance, and because of its continued pervasiveness in industry. The management of PS converting presents a powerful motivation for the pyrometallurgical community to expand its mathematical abilities beyond the confines of matrix algebra and calculus.

This document emphasizes a particular kind of mathematical programming: mixed integer linear programming (MILP), described in Appendix B. Chapters 2-4 adapt the general MILP paradigm to the context of Peirce-Smith converting. But as demonstrated in Chapter 4, this adaptation requires some degree of simplification, i.e. linearization. Chapter 5 demonstrates sample calculations for simplified instances, and illustrates the type of numerical results that can be provided by the MILP framework. Finally, Chapter 6 describes the inherent nonlinearity of the PS converting, and discusses future work in algorithm development.

The current work has resulted in the first MILP framework to manage PS converting systems. It offers a degree of abstraction and adaptability that has not been seen by the extractive metallurgical community at large. This framework is now the backbone of new methodologies that will improve existing operations, and the design of new plants. By considering the defining features of Peirce-Smith converting and its broader industrial characteristics, this work will incite new lines of multidisciplinary research in engineering, applied mathematics and related fields.

1.2 Overview of Extractive Metallurgy

1.2.1 Mineral Concentration

Extractive metallurgy is the set of separation and extraction processes, leading to the production and refinement of metals. Some of these processes are performed at mine sites, and are collectively referred to as mineral concentration.

Within a given mine site, these concentration processes are housed in an industrial building called a concentrator, which produces one or more concentrates. As depicted in Figure 1.1, the concentrates are sent to extractive metallurgical plants, where the metals are chemically extracted.

Some of the extractive metallurgy is performed at the mine sites, within the concentrators, and some is performed in the extractive metallurgical plant. The division between mineral concentration and the subsequent extraction varies from metal to metal, and depends on the nature of the ore that is being mined. Generally, an extractive plant is a logistical hub to a network of mine sites, but may be integrated with a local mine, and with railroad or port facilities.

Ores can be either rocky or clay-like. Rocky ores include copper and nickel sulfides, as well as iron oxides. (Iron sulfides are geologically common, but they are not economically viable as ore because iron oxides are cheaper to process and are sufficiently abundant). Clay-like ores include nickel laterites and aluminum lateritic bauxites.

To process rocky ores, concentrators include grinders to break the ore into fine particles that are amenable to further processing; such concentrators are often called “mills”. To process clay ores, concentrators include roasting ovens to disassemble hydrate complexes and to eliminate volatile species. Both types of concentrators apply separation techniques to divide the valuable minerals from each other, and from the waste gangue.

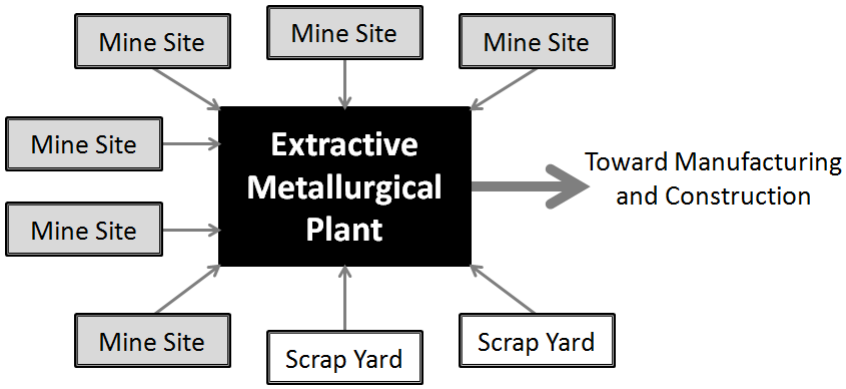


Figure 1.1: Relationship between mine sites, scrap yards and extractive metallurgical plants

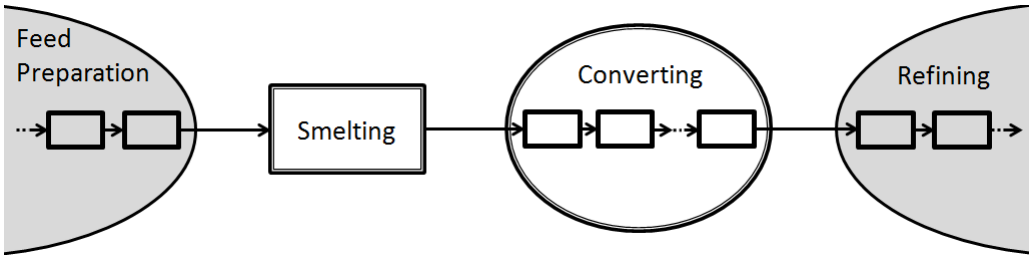
Froth floatation is a particular separation technique in which chemically active bubbles are passed through a slurry, selectively drawing minerals into the skins of the overflowing bubbles[15]. Froth floatation is especially important for sulfide concentration, including copper and nickel-bearing minerals. Other separation techniques used in mineral concentrators may be driven by gravity or by magnetic phenomena.

Concentrates are the primary feed for extractive metallurgical plants. But many of these plants accept recycling scrap as secondary feed. The metal products are consumed within the manufacturing and construction industries (Figure 1.1). Some of these products are sent for additional metallurgical treatments, including additional refining and alloying, as well as deformation, heat and surface treatments. Some of the byproducts of extractive metallurgy are sent to chemical industries, which may actually include other metallurgical sectors.

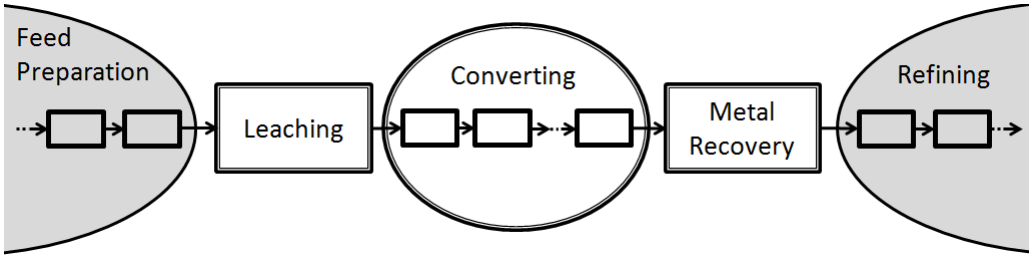
1.2.2 Pyrometallurgical and Hydrometallurgical Extraction

There are traditionally two approaches to extract the valuable metals from concentrates. Firstly, pyrometallurgical extraction is the release of metals via high-temperature reactions occurring in molten or vaporous process streams. Secondly, hydrometallurgical extraction is the release of metals via dissolution reactions occurring in aqueous process streams. Figure 1.2 gives an overview of these two traditional approaches.

The feed preparation often includes drying and/or roasting. Concentrates are normally shipped as wet powder to avoid ignition during transportation. After arrival at the extractive plant, the concentrates may be heat-dried so as to favour the subsequent reactions. The term "roasting" is applied when the amount of heat exceeds that of drying, but is insufficient to cause bulk melting of the concentrate.



(a) Pyrometallurgical extraction



(b) Hydrometallurgical extraction

Figure 1.2: Stages of pyrometallurgical and hydrometallurgical extraction

A roasted concentrate is called a calcine. Certain oxidation or reduction reactions may be performed during roasting. In some pyrometallurgical processes, surface melting is allowed to occur, causing the concentrate to fuse into larger particles or pellets; such calcines are often called sinters [15].

A pyrometallurgical extraction plant is commonly referred to as a smelter. These plants are regarded for their high reaction rates and cost-effectiveness, particularly when dealing with high-grade concentrates, and consumer scrap. Hydrometallurgical plants generally have slower reaction rates than smelters but, for the processing of complex and low-grade mineral concentrates, hydro plants are usually more cost-effective and less polluting [16]. Hydrometallurgical plants are also known for the high-purity of their product.

There are two central operations within a smelter: smelting and converting (Figure 1.2a). Etymologically, "smelting" has the same germanic origins as "melting"[17]. The former has come to imply the complete melting and partial reaction of metal-bearing solids; occasionally, it can imply the evaporation (fuming) of a metal-bearing solid. Once in a fluid state, the smelted material is vulnerable to the intense chemical transformations that constitute converting, and the subsequent reactions that are less intense and which constitute refining.

Within a hydrometallurgical plant, there are three central operations: leaching, converting

and metal recovery (Figure 1.2b). These hydrometallurgical processes are often performed at high pressure. Solid feeds are dissolved in leaching solution (usually acid), and the constituent elements become mobile ions within a pregnant leaching solution (PLS), i.e. a leaching solution that is "pregnant" with metal-bearing content. Once the PLS has been separated from the undissolved solids, it is susceptible to bulk converting reactions. Leaching is indeed the hydrometallurgical equivalent of smelting.

Bulk reactions are only possible if the reactant species are sufficiently mobile; the species may be molten/vapourous as in pyrometallurgical converting, or they may be dissolved as in hydrometallurgical converting. Without sufficient mobility, the rapid reactions are limited to the surfaces, and the bulk of the feed reacts very slowly if at all. Smelting and leaching initiate the metal-bearing material for bulk pyrometallurgical and for bulk hydrometallurgical reactions, respectively.

PS converting is the most common type of converting in copper and nickel smelters [1, 2, 3]. Its main feed is molten furnace matte. "Furnace" refers to the smelting furnaces that precede the converters, and "matte" implies a sulfide phase. PS converting is a type of Bessemerization, which is the forced oxygenation of a molten metal-bearing stream (See Subsection 1.3.1). Essentially, the furnace matte is "converted" into a dense metal-rich fluid, a light slag fluid, and a SO_2-bearing offgas.

The most common type of hydrometallurgical converting within the copper and nickel industry is solvent extraction (SX), [18, 19]. Solvent extraction is a process by which the PLS is mixed with an organic liquid that preferentially absorbs certain ionic species; the organic solution is not soluble in the leaching solution, and eventually disengages from it, carrying away the absorbed species. Thus the initial PLS is "converted" into separate aqueous and organic phases.

SX is the main alternative of PS converting. Although PS converting has pyrometellurgical alternatives as well, which are distinguished through their vessel geometries, and their various mechanisms for receiving feed and discharging products; these topics are discussed in Subsections 1.3.3 and Appendix A.5.

A final pyrometallurgical conversion gives a molten metal product, which is usually subject to further refining (Figure 1.2a). The initial refining stages are sometimes applied directly to the molten metal, which is pyrometallurgical refining. This is often called "fire refining", particularly in copper production [15]. Other refining treatments are applied after the metal has been cast or granulated into solid form; such treatments may be performed in an installation

separate from the smelter, i.e. a metal refinery.

The division between pyrometallurgical converting and pyrometallurgical refining is not always clear. When the feed is highly metallic, either "converting" or "refining" might be the appropriate term, depending on the context. Loosely speaking, converting reactions are deemed to be more intense than refining reactions.

A final hydrometallurgical conversion gives a final pregnant solution. Thus, an additional step is necessary in order to recover the valuable metal(s) from the solution, (Figure 1.2b). This is sometimes accomplished through the application of an electric current (electrowinning, EW), or by manipulating the composition, temperature and pressure to reduce the metal solubilities (precipitation) [16]. The resulting metal is often sufficiently pure not to require refinement, except in cases where ultra-purity is desired.

The metal recovery stage places a clear division between hydrometallurgical converting and the subsequent refining (Figure 1.2b). This division is not present in the pyrometallurgical approach (Figure 1.2a), hence the vagueness between pyrometallurgical converting and refining.

Figure 1.2 is a rather crude summary of pyro- and hydro-metallurgical extraction, which suits the purposes of this document. Actual plant designs feature feedback cycles, scavenging of waste-streams, etc. and varying degrees of redundancy. A more advanced treatment would explore these secondary features.

1.2.3 Hybridization of Pyro- and Hydrometallurgical Extraction

Pyro- and/or hydrometallurgical extraction methods are sometimes hybridized. Technically, these hybrid processes include a partial metallurgical conversion as part of the feed preparation for a subsequent metallurgical extraction.

One reason for hybridization is to implement the respective advantages of pyro- and of hydro-metallurgical extraction. Pyrometallurgical smelting and converting may be succeeded by hydrometallurgical dissolution and electrowinning, thus combining pyro-intensity and hydro-purity; this is typical of nickel smelters [3, 16]. In other cases, a hydrometallurgical approach is used to simplify a complex feed prior smelting, e.g. the simplification of bauxite ores in preparation for aluminum production [20]. As long as there is a smelting operation, the extractive plant may still be referred to as a smelter.

Another reason for hybridization is to assist in stream division. For example, certain pyro

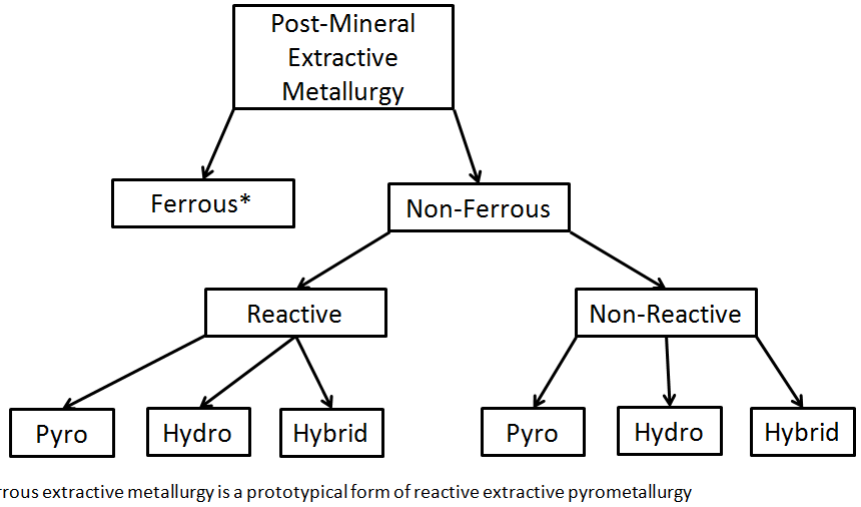


Figure 1.3: Classification of post-mineral extractive metallurgical processes

streams are solidified and granulated in order to apply froth floatation, thus creating several outgoing streams; to continue the conversion, it is then necessary to reactivate the streams, either by heating or dissolving. This is done in certain nickel sulfide smelters that produce copper and cobalt byproducts [3, 16].

In the coming decades, there is likely to be an increasing degree of hybridization within the metallurgical industry. The simple and high-grade ores are being depleted, and the traditional approaches are not as successful on the remaining ores [21]. This continues to motivate new approaches that are optimized for complex and low-grade ores.

1.2.4 Further Divisions within Extractive Metallurgy

Distinctions have been made between mineral concentration, and the subsequent pyrometallurgical, hydrometallurgical and hybrid extraction processes. There are further divisions in extractive metallurgy, namely ferrous vs non-ferrous, and reactive vs non-reactive (Figure 1.3).

Engineering literature and academic curricula treat ferrous metallurgy independently from the rest of metal production because of the pervasiveness of iron in nature, and the relative ease with which it can be extracted. Only the densest deposits of iron oxide are worth exploiting, rendering iron sulfides and other iron compounds undesirable.

Iron is actually the second most common metal in the Earth's crust, yet it represents 95% of the world's metal production tonnage [21], largely because of the form and concentration

in which it is found. Aluminum is more common in nature than iron and has a higher strength-to-weight ratio, but it is far more expensive than iron. Iron has thus evolved to be the multipurpose metal used throughout construction and manufacturing, whereas the non-ferrous metals are selected only when their special properties are essential.

In themselves, iron and steel alloys can fill a wide range of manufacturing demands via the addition and removal of alloying elements, as well as surface and thermal treatments. In these cases, iron is the main component, and it is often small amounts of non-ferrous metals that bring forth the special properties of the given alloy. Indeed, much of non-ferrous metal consumption is in steel alloys; for example, 60% of the global nickel production is intended for stainless steel and related alloys [22].

Historically, there has been considerable adaptation of concepts and technology from ferrometallurgical extraction into non-ferrous extraction. The influence has been in both reactive and non-reactive processing, and especially in the pyrometallurgical branches. Peirce-Smith converting is one example of non-ferrous technology that evolved from ferrous technology (See Section 1.3.1).

Figure 1.3 makes a distinction between the extraction of reactive metals and non-reactive metals, which are each propagated into pyro, hydro and hybrid processes. In this context, "reactive" specifically implies the tendency to form oxides when smelted or leached in regular air environments [23]. Metals such as copper, nickel and cobalt may be smelted/leached without pervasive oxidation, and are hence non-reactive, likewise for gold, silver and other noble metals. Aluminum, magnesium, zinc, tin and the like require special equipment or processing to protect them from air environments during smelting/leaching.

The traditional way to extract a reactive metal from a concentrate is to pyrometallurgically reduce the oxide concentrate or calcine in the presence of carbon (carbothermic reduction); this is still widely practiced for iron and tin. This is also common for zinc production, although 80% of primary zinc is now extracted through hydrometallurgical means [24]. In this context "reduce" implies the introduction or restoration of electrons to the metal atoms which, in chemistry jargon, is the opposite of "oxidize".

Incidentally, the treatment of iron oxides employs coke (roasted coal) to carry away the bonded oxygen. Ferrous extractive metallurgy is therefore a form of reactive pyrometallurgy. The carbothermic reduction of iron oxides results in a carbon-saturated molten metal, known as pig iron. The conversion of pig iron into steel is similar to PS converting, as discussed in Subsection 1.3.1.

Some reactive metals are too reactive for carbothermic reduction, meaning that the ionic bond to oxygen is so stabilizing that uneconomical quantities of heat are needed to process it in the traditional way. For example, alumina Al_2O_3 is such a stable oxide, with such a high melting point (2054°C), that it is not economical to reduce Al_2O_3 by carbothermic means, as in ironmaking (Fe_2O_3 melts at 1370°C). Metallurgists have found other ways of managing and delivering the energy required to release these bonded metals.

Aluminum smelters do not melt alumina in the classical sense. Rather, the alumina is dissolved in a bath of molten cryolite Na_3AlF_6 held at roughly 1000°C. The dissolved aluminum ions are reduced into metal as electricity is passed from a consumable graphite (carbon) anode, through the melt, and into the outer graphite shell that acts as the cathode. This constitutes the Hall-Heroult process [20], and is a form of carbo-electrothermal reduction. The liquid aluminum is heavier than the cryolite solution, and is therefore drained from the bottom of the cell.

Aluminum is the prime example of a reactive metal where the traditional carbo-reduction is inadequate. It is Nature's most abundant metal, yet prior to the Hall-Heroux process (1886), it had been more valuable than gold [25]. There are also other reactive metals, such as magnesium and titanium, which form sizable portions of the terran crust, yet their cost precludes them from prominence in manufacturing and construction.

Copper, nickel and other non-reactive metals have a tremendous advantage over reactive metals. They can be processed as molten matte [15]. Sulfur and iron may be carried away by blasting an oxidizing stream through the matte, which is the principle behind Peirce-Smith converting and related technology.

Generally in extractive metallurgy, there is a technological challenge to render the metal and its bonding elements (sulphur, iron, oxygen, chlorine, etc.) sufficiently mobile to permit separation [15, 16]. The problem is more complicated for reactive metals, especially when the bonds are too strong for carbothermal reduction. The rest of this document focuses on copper and nickel extraction, which are non-reactive metals, and thus do not suffer from these complications.

1.2.5 Extraction of Copper and Nickel

The world primary copper supply consists almost entirely of sulfide ores, rich in chalcopyrite $CuFeS_2$. The nickel supply is divided evenly between sulfides rich in pentlandite $(Fe,Ni)_9S_8$, and laterites rich in oxides and silicates [2, 3]. In the coming years it is expected

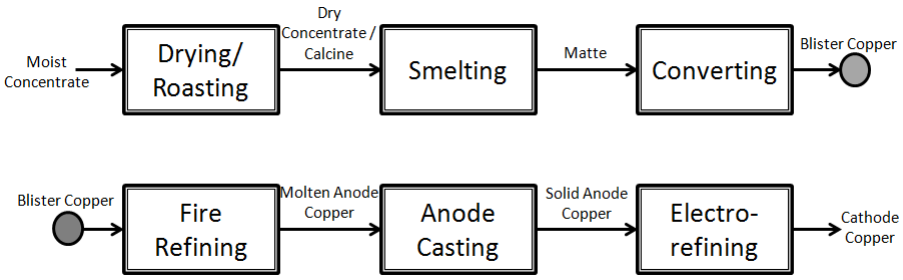


Figure 1.4: Flow diagram for a conventional copper smelter

that the laterite-based nickel production will exceed the sulfide-based nickel production [2].

Within a concentrator, copper and nickel sulfide ores undergo milling and froth floatation. On the other hand, nickel laterites undergo reduction roasting. Various pyro- and hydrometallurgical extractions are available for each of these ore classes.

Figure 1.4 depicts a typical flow diagram for a conventional copper smelter. Roasting is not necessary in modern smelters that employ flash smelting [26]; this entails a special furnace which allows the roasting reactions to occur while the concentrate powder is falling through the feed chute. Roasting is still performed as feed preparation for electric smelting, and certain antiquated smelting technologies[15].

Smelting is succeeded by converting (usually Peirce-Smith) which eliminates iron and sulfur, and gives a crude form of metallic copper, known as "blister copper", or simply "blister"[7]. It is roughly 99 wt%Cu with 1 wt%S. Blister copper is not allowed to solidify, otherwise the residual sulfur is expelled from the cooling metal, resulting in SO_2 blisters, hence the name. To prevent the formation of blisters, the residual sulfur and oxygen is removed in the fire refining stage which immediately follows the converting. The term "matte" does not apply to blister copper because it is predominantly metallic, as opposed to sulfide.

Fire refining furnaces are sometimes called anode furnaces, because they produce a copper that is sufficiently pure to be cast into anodes (roughly 99.5 wt%Cu), that are then subject to electrorefining. The final cathode is roughly 99.99 wt%Cu [15].

Most copper production occurs through the pyrometallurgical route (Figure 1.4). But following the 1970's, a purely hydrometallurgical extraction accounts for roughly 20% of primary copper production, consisting of leaching-SX-EW [18]. This approach is appropriate for oxide ores, and low-grade sulfide ores that contain some degree of copper oxide. This hydro approach consumes considerable amounts of sulfuric acid, which is a byproduct of

pyrometallurgical copper production. For this reason, leaching-SX-EW plants are often built alongside copper smelters.

Nickel sulfide concentrates are often rich in copper and cobalt sulfides, as well as iron sulfide. As with copper, the conventional extraction techniques begin with smelting and converting (usually Peirce-Smith), which eliminates the iron sulfide. The conversion of nickel matte results in a nearly iron-free substance, known as "converter matte" or sometimes as "Bessemer matte" [3, 15]. In this case, "matte" is fitting because the product is composed of nickel sulfide, and often a considerable amount copper and cobalt sulfide.

Direct-Outotec-Nickel (DON) technology combines smelting and converting into a single continuous operation [26], thus the mineral concentrate is converted directly into converter matte. This process is employed in two installations: Harjavalta Oy (Finland) and Fortaleza (Brazil). In the Harjavalta Oy plant, DON operates in parallel to a conventional copper smelting line (Figure 4). The Harjavalta Oy plant has DON for nickel extraction, and Peirce-Smith Converting for copper extraction.

There is considerable variation in the methods for treating Ni-bearing converter matte. One approach involves the passing of carbon monoxide through solidified converter matte, thus carrying away the nickel as a cabonyl vapour [15].

One hybrid approach for nickel extraction is given by smelting-converting-SC-leaching-SX-EW [9, 16]; "SC" refers to a slow-cooling technique that forms solid granules. When there is high copper and PGM content, the slow-cooling may be followed by grinding and froth floatation (FF) in order to isolate the sulfides into separate streams, thus giving smelting-converting-SC-grinding-FF-leaching-SX-EW [9, 15, 27]. If a sufficiently low iron content is attained in the converter matte, it is possible to cast a sulfide anode, hence smelting-converting-casting-EW, or smelting-converting-SC-grinding-FF-casting-EW [9, 15]. Also, the electrowinning can be substituted by hydrogen reduction, a form of precipitation [19].

A purely hydrometallurgical extraction (leaching-SX-EW) has recently been implemented to treat nickel sulfides as part of the Voisey's Bay project [28]. There are in fact three solvent extraction stages which act in sequence, sending into the organic phase (1) copper, (2) impurities and (3) cobalt; hence the nickel remains in the final aqueous phase. The three metal-bearing streams all undergo EW.

Most of the nickel laterites undergo pyrometallurgical processing to produce ferronickel FeNi, which is used as an alloying agent in stainless steel. However, there are two laterite

smelters, Doniambo (New Caledonia) and Sorowako (Indonesia), that employ PS converting to obtain a nickel-sulfide matte, which is subject to EW for nickel metal recovery [2].

The remaining laterites undergo hydrometallurgical extraction to release metallic nickel; cobalt is often a major byproduct. Some plants employ high-pressure-acid-leaching followed by SX-EW, while others use an ammonical leach instead of an acid leach [19].

It should be noted that copper and nickel ores are often rich in silver, gold and platinum group metals (PGM). These precious metals are usually recovered as byproducts during metallurgical refining, or posttreatment of electrowinning solution. There are smelters in South Africa that treat nickel sulfides that are so laden with PGM that the main value is from latter [3]; the nickel is hence a byproduct of PGM extraction.

Peirce-Smith converting has a central role in copper and nickel extraction from sulfide ores, administering much of the bulk reactions. Secondly, for the processing of nickel laterite ores, PS performs a conversion that ultimately leads to nickel metal rather than ferronickel. Yet there are hydrometallurgical plants, which circumnavigate PS using SX. These are in addition to the pyrometallurgical alternatives to PS converting (Appendix A.5). For existing copper and nickel smelters to remain competitive with newer plants, continued improvements to PS converting will be essential.

1.3 Overview of PS Converting

1.3.1 PS Converting as a Bessemerization Process

In copper PS, the furnace matte is mainly a mixture of copper sulfide and iron sulfide, which is converted into blister copper. In nickel PS, the furnace matte is generally a mixture of nickel sulfide, iron sulfide, copper sulfide and cobalt sulfide, which is converted into Bessemer matte. In both cases, the conversion involves the pneumatically forced oxygenation of the matte, which eliminates iron and sulfur atoms.

The pneumatic oxygenation is occasionally referred to as Bessemerization (Figure 1.5),

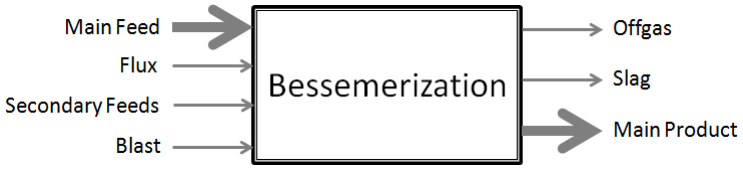


Figure 1.5: Input and output streams of a Bessemerization process

alluding to Henry Bessemer, the father of modern steelmaking. In 1856, Bessemer developed a process which blasts oxygen through carbon-saturated pig iron, resulting in a raw form of steel; PS converting was patented in 1909 by Peirce and Smith, well after the Bessemer process [29]. In the early literature, including the original patent [30], PS converting is described as the Bessemerization of matte. More recent times, the term "converting" is synonymous to Bessemermizing, at least in the context of iron, copper and nickel extraction.

The modern Bessemerization of iron is performed using a so-called basic oxygen furnace (BOF). In this context "basic" refers to the alkalinity of the refractory bricks that line the BOF [31]. (PS converters are also lined with basic refractories, as discussed in Subsection 1.3.3 and Appendices A.1). The Bessemerization of copper and nickel mattes is performed mainly by PS, although there are a few alternatives (Appendix A.5).

A properly conceived Bessemerization process leads to oxidized byproducts that are fluid (liquid and gas), hence easily separated from the main product. The liquid byproducts form a light slag that floats over the main product, and can eventually be skimmed off. The gaseous byproducts leave the system as offgas. A PS offgas is a mixture of SO_2 and nitrogen, while a BOF offgas is a mixture of CO and CO_2.

The composition of the slag is carefully controlled through the addition of flux, i.e. particulate solid feed, usually with diameters no larger than 5 cm [7, 9]. The composition is determined in relation to the operating temperature, so that the slag is thermodynamically prone to capturing target chemical elements, while maintaining sufficiently low density and viscosity. The low density ensures that the slag floats above the rest of the melt. The low viscosity ensures that the offgas can bubble through the slag without excessive resistance, and that the slag can be evenly pored (skimmed) out of the vessel with minimal entrainment of the main product.

Fluxes generally contain stable oxides, such as SiO_2, CaO, Al_2O_3 and MgO. In copper and nickel matte conversion, the main fluxing agent is silica SiO_2, which contributes to the iron-silicate slag. Burnt lime (CaO) or dolomite ($CaMg(CO_3)_2$) is used in steelmaking because it draws away phosphorous and other impurities, which would embrittle or otherwise deteriorate the steel [31].

Bessemerization processes can include secondary feeds that have a character similar to either the flux, the main feed, or a mixture of the flux and main feed. In Peirce-Smith converting, the secondary feeds could include copper scrap or recuperated flue dust, for example. The excess heat of PS converting allows cold secondary feeds to be melted down

and blended with the rest of the charge, so that their valuable content can be recovered. This is discussed further in Subsections 1.3.2 and 1.3.3.

The oxygen-bearing stream of a Bessemerization process is called the blast. Traditionally, the blast consisted of compressed air (21 vol%O_2). In modern PS operations, it is now common to use an oxygen-enriched blast, usually 25-28 vol%O_2[1] which increases the productivity of the blast, and has other benefits that will be discussed in Subsection 1.3.3. Other processes employ much higher enrichment; for instance, BOF employs a blast that is commercial grade oxygen, over 99 vol%O_2 [31].

PS cannot attain the same blast enrichment as BOF due to a difference in the delivery mechanism. PS is side-blown through a system of tuyeres that penetrate the refractory lining; too much oxygen will cause locally intense reactions and rampant abrasion of the lining. In a BOF, the blast is usually introduced through a single lance that hangs deep into the center of the furnace, away from the refractory lining. In any case, a higher enrichment is not necessarily practical in a PS converter, due to the possibility of overheating (See Subsection 1.3.3).

BOF blast rates usually range from 30000 to 60000 Nm^3/h [31], and similarly for copper PS [7]. In nickel PS it can be much lower, usually between 5000 and 25000 Nm^3/h depending on the grade of the matte [3]. (The units Nm^3/h are described below). Especially in combined Ni-PGM production, a low blast rate allows better control of the reactions and a higher recovery of the precious metals. It should be noted that an increase in O_2 enrichment can be accompanied by a decrease in blast rate, to maintain the same production rate.

By convention, blast rates are measured in "Normal meters cubed per hour" (Nm^3/hr), or sometimes "Normal meters cubed per minute" (Nm^3/min). This implies that the blast rates are tabulated as if they were under Normal Conditions, meaning atmospheric pressure and a temperature of 0°C. For instance, 25000 Nm^3 of O_2 is the mass of oxygen gas that, under Normal Conditions, occupies a volume of 25000 m^3. The adherence to Normal Conditions provides a means to compare gas flow rates, independent of the compression.

In summary, the molten material is subject to forced oxygenation and the addition of oxide flux. Table 1.1 compares the three Bessemerization Processes.

Table 1.1: Summary of Bessemerization processes

	Basic Oxygen Process	Cu PS Converting	Ni PS Converting
Main Feed	pig iron	Fe/Cu matte	Fe/Ni/Co/Cu matte
Flux	burnt lime or dolomite	silica (mainly)	silica (mainly)
O_2 Enrichment	$> 99\%$	21-28%	21-28%
Blast (Nm^3/hr)	30000 - 60000	30000 - 60000	5000-25000
Main Product	steel	blister copper	Ni/Co/Cu matte
Slag	mixed oxide, high in CaO	iron-silicate	iron-silicate
Offgas	CO/CO_2	N_2/SO_2	N_2/SO_2

1.3.2 Matte Converting Reactions

Copper matte is essentially a mixture of FeS and Cu_2S, and undergoes two stages of converting, firstly the Slag-Blow which eliminates the FeS, and secondly the Copper-Blow which releases the blister copper. Nickel matte is generally described as FeS, Ni_3S_2, Cu_2S and CoS. Nickel PS consists of only the Slag-Blow stage, which eliminates the FeS.

The Slag-Blow is described generally by:

$$[\mathrm{FeS}_{(\mathrm{liq})}]_{\mathrm{Matte}} + [\mathrm{O}_{2(\mathrm{gas})}]_{\mathrm{Blast}} + \mathrm{Flux} \rightarrow [\mathrm{FeO}_x{\cdot}\mathrm{Flux}_{(\mathrm{liq/sol})}]_{\mathrm{Slag}} + [\mathrm{SO}_{2(\mathrm{gas})}]_{\mathrm{Offgas}}$$

Thus the Slag-Blow results in a slag stream. For the sake of generality, this expression has not been stoichiometrically balanced. In the most common case, the flux is composed entirely of SiO_2, and the resulting slag is fayalite Fe_2SiO_4 with 5 wt% to 15 wt% magnetite Fe_3O_4. Accordingly, the Slag-Blow is a combination of two reactions [7, 15]:

$$2[\mathrm{FeS}_{(\mathrm{liq})}]_{\mathrm{Matte}} + 2[\mathrm{O}_{2(\mathrm{gas})}]_{\mathrm{Blast}} + [\mathrm{SiO}_{2(\mathrm{sol})}]_{\mathrm{Flux}} \rightarrow [\mathrm{Fe}_2\mathrm{SiO}_{4(\mathrm{liq})}]_{\mathrm{Slag}} + 2[\mathrm{SO}_{2(\mathrm{gas})}]_{\mathrm{Offgas}}$$

and

$$3[\mathrm{FeS}_{(\mathrm{liq})}]_{\mathrm{Matte}} + 5[\mathrm{O}_{2(\mathrm{gas})}]_{\mathrm{Blast}} \rightarrow [\mathrm{Fe}_3\mathrm{O}_{4(\mathrm{sol})}]_{\mathrm{Slag}} + 3[\mathrm{SO}_{2(\mathrm{gas})}]_{\mathrm{Offgas}}$$

Other stable oxides such as CaO, Al_2O_3 and MgO are often included with the SiO_2 to form an olivine slag instead of the classic fayalite; this type of blending has been observed to minimize the copper losses in the slag, to decrease the amount of magnetite in the slag, and to diminish the corrosion of the refractory [32].

Within the ferrous component of the slag, fayalite Fe_2SiO_4, which is often denoted as $2FeO{\cdot}SiO_2$, is preferable to magnetite because it binds one atom of blast oxygen to every iron atom. Magnetite Fe_3O_4 represents a less efficient use of the blast, binding 4/3 oxygens for

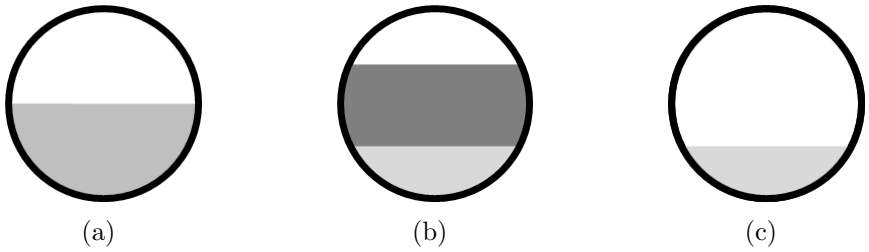


Figure 1.6: Bath volume during Slag-Blow stage (a) prior to blow, (b) after blow, (c) after skimming

every iron. Also fayalite is quicker to rise from the matte since it is liquid, whereas magnetite is solid.

As a Slag-Blow operation is performed within a vessel, there is a danger of overflow, hence an operational constraint. The matte volume decreases as the FeS is reacted away, but this is more than offset by the increase in slag volume (Figure 1.6). Indeed, $(FeO_x \cdot Flux)_{Slag}$ is less dense than FeS in terms of mass/volume, but also in terms of (mass of Fe)/volume. Typical density values for slag and matte are 3.5 T/m^3 and 5.5 T/m^3, respectively [33]. Eventually the slag must be skimmed away, to allow more feed and/or more blowing.

There is also a danger of overheating, because the Slag-Blow reactions are exothermic. To maintain the target operating temperature (1200-1250°C), the molten matte must be supplemented with cold charge. This may include recycled material from other parts of the smelter, or scrap metal, or concentrate [15].

Through the elimination of FeS, the Slag-Blow reaction lowers the iron content of the matte to roughly 1 wt%; the remaining matte is the so-called Bessemer matte, converter matte, or "white metal"[15]. For Nickel PS, the Bessemer matte is the main output; the Ni_3S_2, CoS and Cu_2S are separated and treated in the subsequent processes that were introduced in Subsection 1.2.5.

In copper PS, the converter matte is most commonly called "white metal", even though it is only semi-metallic [15]. This material is composed of Cu_2S and is subject to the Copper-Blow, as described by

$$[Cu_2S_{(liq)}]_{Matte} + [O_{2(gas)}]_{Blast} \rightarrow 2[Cu_{(liq)}]_{Blister} + [SO_{2(gas)}]_{Offgas}$$

This results in blister copper, saturated in sulfur. It is sent directly to the fire refining furnace

Table 1.2: Species present in matte, in increasing order of thermodynamic stability

	FeS	Ni_3S_2	CoS	Cu_2S
Cu-bearing matte	√	×	×	√
Ni-bearing matte	√	√	√	√

so that the residual sulfur can be removed prior to casting. This reaction does not require flux, nor does it produce slag.

Blister copper is more dense than white metal, its parent phase. Therefore, it tends to form at the bottom of the vessel. Secondly, the conversion of white metal (5.2 T/m^3) into blister copper (8.0 T/m^3) signifies a net decrease in volume, so there is no danger of overflow [34]. However, there is a danger of overheating because the reaction is exothermic. So there is again a need for cold charges. In this case, the only acceptable cold charges are those which consist of copper, sulfur, oxygen and only trace amounts of other elements. The other elements are likely to interfere with the release of copper; for instance, any iron-bearing species would cause the Slag-Blow to resume.

Of all the sulfides under consideration (Table 1.2), Cu_2S is the only one that releases metal under PS temperatures ($1200 - 1250°C$); the oxygen bonds only to the S in this case, leaving the Cu to form the blister copper phase [7]. For the non-cuprous sulfides, the oxygen bonds to both the metal and the sulfur atoms; in the case of FeS, for example, the oxygen bonds to the Fe as well as the S, which precludes the emergence of a liquid iron product. Similarly, liquid nickel and cobalt products are also precluded.

It is possible to release liquid nickel and/or cobalt, but this route is not supported by PS because this demands temperatures above 1455°C, the melting point of nickel [15]. On the other hand, liquid iron is not released unless a reductant is introduced. (As discussed in Subsection 1.2.4, ferrous metallurgy can be regarded as a type of reactive metallurgy).

Overblowing occurs in PS, when the bulk of the valuable metals begin to oxidize and form slag. Overblowing should not be practiced in the presence of iron-bearing slag because it is costly to separate the valuable oxides from the undesirable iron oxides [35]. After the iron has been skimmed away, however, some degree of overblowing is often practiced to assure complete conversion throughout the final blister copper or the Ni converter matte.

In copper PS, overblowing helps diminish the residual sulfur, which decreases the need for fire refining [36, 37]; it can also be used to control minor elements such as lead and bismuth

[37]. The Overblow reaction for copper PS is

$$4[Cu_{(liq)}]_{Blister} + [O_{2(gas)}]_{Blast} \rightarrow 2[Cu_2O_{(sol)}]_{Slag}$$

The resulting copper-oxide slag is then recycled into other parts of the smelter. Thus, there is an optimal circulating load that is determined on a tactical level.

In nickel PS, overblowing occurs once the Slag-Blow reaction has been completed. The least stable of the non-ferrous sulfides is Ni_3S_2, so the Slag-Blow is followed by the Nickel-Overblow

$$2[Ni_3S_{2(liq)}]_{Matte} + 7[O_{2(gas)}]_{Blast} \rightarrow 6[NiO_{(sol)}]_{Slag} + 4[SO_{2(gas)}]_{Offgas}$$

After (nearly) all of the Ni_3S_2 has been blown out of the matte, the next least stable sulfide is CoS, so the following reaction is the Cobalt-Overblow

$$2[CoS_{(liq)}]_{Matte} + 3[O_{2(gas)}]_{Blast} \rightarrow 2[CoO_{(sol)}]_{Slag} + 2[SO_{2(gas)}]_{Offgas}$$

After (nearly) all of the CoS has been blown out, the only remaining sulfide is Cu_2S, which is akin to the white metal from copper PS. The blowing of white metal is essentially the Copper-Blow, discussed earlier.

If appropriate fire refining equipment is available, it is theoretically possible for nickel smelters to perform a conventional copper extraction, by applying a Copper-Blow followed by the fire refining. Of course, after the Copper-Blow reaction has reached completion, the continued blowing is akin to the copper PS Overblow, a.k.a. the Copper-Overblow.

Given the possibility overflowing, overheating and overoxygenation, Peirce-Smith converting cannot simply be applied haphazardly. The reactions must be timed, and coordinated with the handling of feed and product streams.

1.3.3 PS Converting Technology

The reactions described in the previous section are housed within rotary furnaces known as Peirce-Smith converters (PSC), as depicted in Figure 1.7. Each of these vessels is lined with refractory bricks, that are penetrated by a row of tuyeres for gas injection (Figure 1.8); this area of the converter is called the tuyere-belt. There is a large mouth at the top of the cylindrical drum that is held upright during operation, underneath a fume hood. Other vessels are described in Appendix A.5.

At the beginning of a cycle, the converter is rotated to its forward position, so that it can

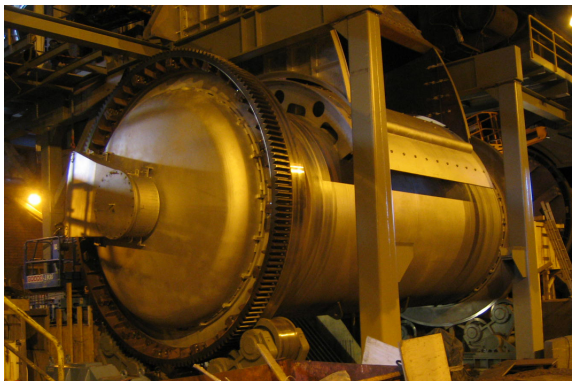
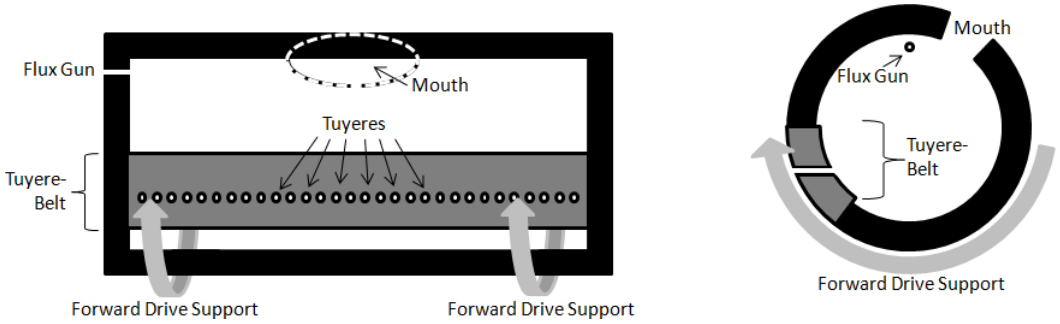


Figure 1.7: Newly commissioned Peirce-Smith converter at the Harjavalta Oy Smelter [38]



(a) Side-view (b) Cross-section

Figure 1.8: Interior of a Peirce-Smith converter

be filled to roughly half of its volume [7, 9, 15]. The converter is then rotated back to its upright position, as the tuyeres are activated. The tuyeres are submerged below the surface of the liquid matte, thus blasting into the matte. Silicious flux is added as needed, combining with the iron and oxygen to form slag; the flux can be introduced either through flux guns or from chutes, as described below.

As the cycle progresses, blowing is halted periodically to remove the accumulated slag, and replace it with fresh feed. The vessel is rotated forward, to pour the slag into a ladle which is then carried away for further processing. The vessel is then rotated back partially, so that it may host one or more ladles of furnace matte and/or other feeds. Once the converter is sufficiently filled, the drum is rotated back into the upright position, and blowing is resumed.

Eventually enough white metal (or converter matte) is accumulated in the vessel, at which point the last of the slag is carefully skimmed away. For copper PS, this is when the copper blow is applied. For nickel PS, there may be a special finishing blow, but this depends on the

nature of the feed, and the nature of the downstream operations. Afterward, the converter product is discharged, thus ending the cycle.

A typical copper PS cycle can last 6 to 12 hours to convert roughly 200 tonnes of furnace matte having 60 %Cu grade [7]; the cycle time includes charging, blowing, skimming, discharging and idle time, and is heavily dependant on the nature and quantity of the secondary feed. There is even more variability within the nickel industry; Kyllo and Richards describe cycles that last roughly 9 hours [39].

Converter dimensions are such that the length is 2 to 2.5 times the diameter, in such sizes as 13 feet in diameter by 30 feet long (3.96 $\times$ 9.15 m), 12 feet in diameter by 28 feet long (3.66 $\times$ 8.55 m), and 13 feet in diameter by 35 feet long (3.96 $\times$ 10.67 m) [15]. In modern smelters there is a trend toward larger vessels, the largest being roughly 15 feet in diameter by 44 feet long [40].

In a typical Peirce-Smith converter, the tuyeres are placed at intervals of 15-40 cm [3, 7] within the tuyere-belt, but they may be omitted from the region surrounding the mouth to prevent the bath from splashing out. Conventional tuyeres are made of steel, 5 cm in diameter, and operated at 80-120 kPa gauge pressure to deliver between 80 and 1000 Nm^3/min [3, 7]. During conventional operation, the entering blast causes a chill so that a "cold nose" solidifies at the tuyere tip. This accretion must be dislodged periodically using a punching machine. However there are now some smelters that can operate at higher pressures ($>$ 250 kPa), which prevent accretion and avoid the need for punching (Appendix A.2).

During the Slag-Blow, powdered flux is introduced through guns and/or a chute [7, 8, 9]. One or more flux guns (a.k.a. "Garr guns") can be installed at the circular ends of the furnace, to launch powdered flux through a hole that is placed above the slag surface. Chutes are placed above the converter, so as to drop the flux through the mouth of the converter and into the melt. The slag level should remain sufficiently below the flux guns, otherwise the launch trajectory will be hindered, and the slag will lose its desired uniformity.

Also the tuyeres should remain deep within the matte, so that the blast can properly react with the melt and ensure a rigorous mixing [9]. This, along with the uniform distribution of fine flux particles, leads to a high oxygen efficiency; that is the fraction of blast oxygen which reacts with the melt. A well-operated converter attains values ranging between 90 and 95% [7], and the rest of the oxygen passes into the offgas.

There are traditionally two features concerning the refractory lining, which distinguished

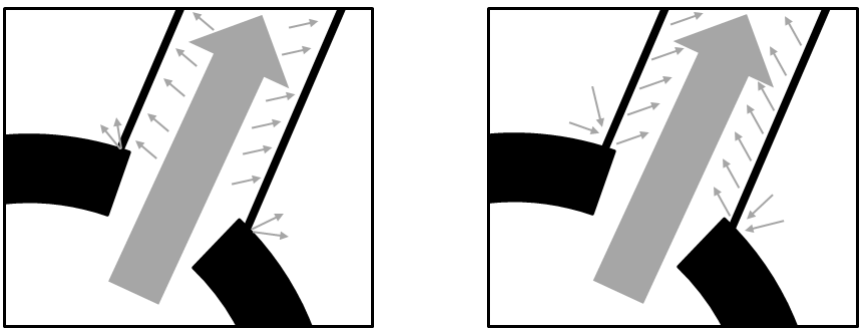
the PSC from its predecessors. Firstly, there is usually a gap between the lining and the outer steel shell [29]; this accommodates the thermal expansion that occurs during operation. (Nonetheless, advances in the refractory materials, have made it possible to eliminate the gap, and thus to minimize the risk of metal leakage [40]).

Secondly, a PS lining is composed of basic refractories [29], i.e. a thermoresistant material that resists reaction to silicates, hydroxides and other negative ions. Originally the lining was made entirely of magnesia (MgO), although the thermomechanical properties have been greatly enhanced by including chromia (C_2O_3) and alumina (Al_2O_3). Today, a typical lining has roughly 60 wt%MgO, 20 wt%C_2O_3, 8 wt%Al_2O_3, 7 wt%FeO, and small amounts of SiO_2, CaO and active metals [41]. The predecessors of PS converting often employed an acid-refractory lining, made of SiO_2 that would be rapidly consumed during the Slag-Blow.

The most vulnerable part of the refractory is the tuyere belt, which undergoes tremendous abrasion and thermal shock. The tuyere-belt can typically withstand 130 cycles before it must be replaced, versus the rest of the lining which can withstand over twice as many cycles [7].

One of the major shortcomings of Peirce-Smith converters concerns the capture of the offgas. The offgas leaves through the mouth, usually carrying oxide fumes, traveling up into a hood and toward the acid plant. Peirce-Smith converters are notorious for releasing SO_2 into the surrounding work area during charging and pouring, whenever the mouth is not aligned with the hood [9, 42]. Also, the gas pressure can occasionally become too high during blowing, thus overwhelming the hood (Figure 1.9a).

More typically, the hood is underwhelmed, and surrounding atmosphere is pulled up into the hood and dilutes the offgas stream (Figure 1.9b). Acid production is only profitable for



(a) Leaking of offgas (b) Infiltration of external air

Figure 1.9: Interaction between offgas and external air

Table 1.3: SO_2 content of offgas streams, calculated as a function of blast enrichment, assuming oxygen efficiencies from 85 to 95%, and dilution factors ranging of 2 to 2.5

	Regular air blast (21 vol%O_2)	Enriched blast (28 vol%O_2)
Slag-Blow	$4.5 - 6.4$ vol%SO_2	$6.2 - 8.8$ vol%SO_2
Copper-Blow	$7.0 - 9.8$ vol%SO_2	$9.4 - 13.1$ vol%SO_2

offgases that contain over 10 vol% SO_2, and the dilution effect makes it difficult for PSC operators obtain this level. To mitigate the dilution problem, it is now common to use a water-cooled hood, which allows a tighter fit, and reduces the deliberate infiltration that had been to prevent overheating of the hood [42].

To address the dilution problem differently, a modified PSC was developed in 1931 in Hoboken, Belgium [15, 43]. The Hoboken converter draws the offgas through a siphon ("goose-neck"), as depicted in Figure 1.10. This allows the converter mouth to be sealed during blowing, hence minimizing the contact between the offgas and the outside air. These units obtain roughly 12 vol%SO_2 in the offgas, which is favorable to acid production [8]. There are currently only a handful of these converters in use [1, 8], possibly because of the accretions that tend to form in the goose-neck [7].

In conventional PS converters, the dilution effect has been mitigated by enriching the blast. Table 1.3 compares post-dilution offgas concentrations for a regular air blast and an oxygen enriched blast, assuming typical dilution factors and oxygen efficiencies. These results demonstrate that oxygen enrichment favours acid production. However, there are two limitations: (1) the supply of cold charge, and (2) the integrity of the tuyere-belt.

The first limitation is described in Figure 1.11. A higher blast enrichment corresponds to a higher proportion of O_2, and thus a lower proportion of N_2. Even though the nitrogen is chemically inert, it is still important as a coolant; it enters the vessel at the blast temperature ($\approx 50°C$), and is exhausted at roughly the bath temperature ($\approx 1250°C$), hence convecting away the heat associated with a $\approx 1200°C$ differential. Therefore a decrease in nitrogen must be compensated with an increase in cold charge.

As for the second limitation to oxygen enrichment, the tuyere-belt refractories disintegrate rapidly when the enrichment is brought beyond 30 vol%O_2. Certain smelters have extended this limit by replacing the traditional tuyeres with Air Liquide Shrouded Injectors (ALSI), first presented in 1995 [44, 45]. This is a system of two concentric pipes, in which pure nitrogen is sent through the outer pipe to form a shroud, as enriched oxygen is sent through

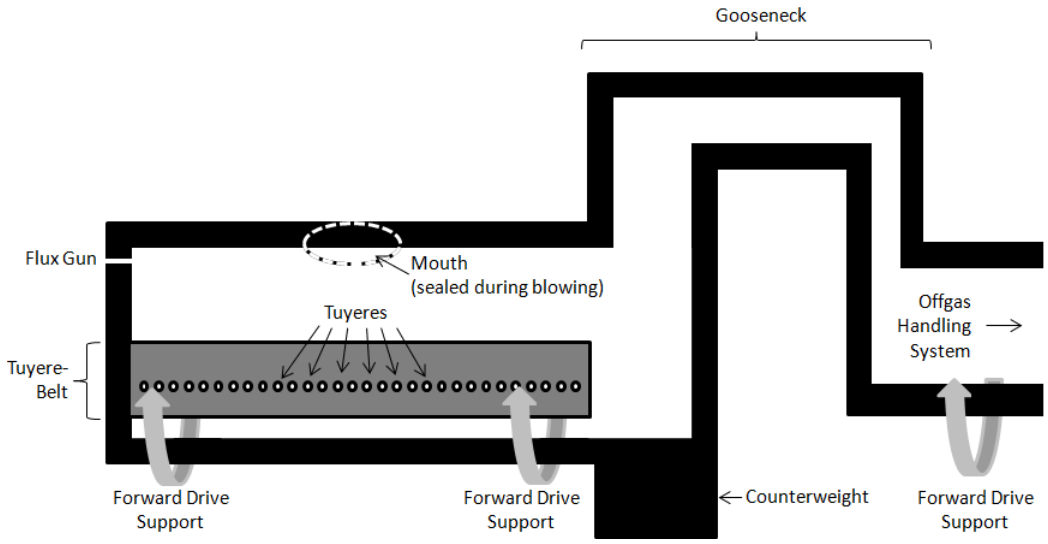


Figure 1.10: Interior of a Hoboken converter (side-view)

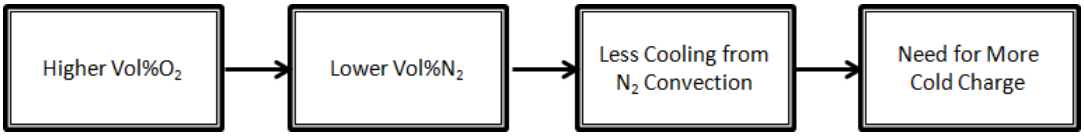


Figure 1.11: Relationship between O_2 enrichment and the demand for cold charge

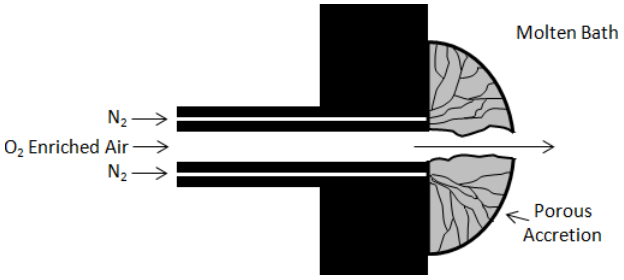


Figure 1.12: ALSI Technology [7]

the inner pipe. Thus the cooling effect is concentrated to the outer ring, forming a protective accretion of solidified porous material (Figure 1.12). This is a more effective distribution of blast N_2 and blast O_2 that allows for a net enrichment of 30-60 vol%O_2. Furthermore, ALSI operates at a sufficiently high pressure that it does not require punching. Unfortunately, ALSI does not perform well below 30 vol%O_2, so there must be a sufficient supply of cold charge. The most notable installation of ALSI is at the Falconbridge Smelter [45, 46], in the so-called "Slag Make Converter", which is essentially a large PSC.

Figure 1.11 explains how a lack of appropriate cold charge can limit the O_2 enrichment. This in turn can limit profitable acid production, in spite of the advancements in air-injection technology. Offgas treatment is further complicated by the fact that PS and Hoboken converting are batch processes, which send uneven loads to the acid plant; this is one of the motivations for continuous converting, discussed in Appendix A.5.

Regardless of its limitations, PS has remained the workhorse of the copper and nickel industries for over a century. Since that time, there continue to be incremental improvements in the technology which are carefully marketed to existing smelters and to the builders of new smelters. To evaluate the impact of an enhanced PSC, there is a need to simulate its interaction with other components of the smelter, including the other PS converters.

1.3.4 The PS Converter Problem

Normally there are two to five Peirce-Smith converters operating in parallel within a copper smelter [7], and three to six within a nickel smelter [3], sharing ancillary objects such as ladles, cranes and offgas launders. This system is commonly referred to as a converting aisle, because of how the converters are arranged within a plant, side-by-side (Figure 1.13).

The Peirce-Smith Converter Problem is to coordinate Peirce-Smith (or Hoboken) converters with other objects within the system, so as to maximize a production measure within a fixed period of time, while respecting chemical, volumetric and thermal constraints.

Suppose a plant manager wishes to upgrade the offgas treatment facility so that they can run more PS converters simultaneously without violating government-imposed SO_2 limits. After the upgrade, they may find that there is now a shortage of feed ladles, which hinders the full benefit of the upgrade. It may take an additional $100 000 to buy new ladles and to install a new crane system, etc. After this additional upgrade, they may find that there is now a shortage of cold feed, which must now be shared between a larger number of converters.

The general configuration of the converter aisle can vary from smelter to smelter. Also,

Figure 1.13: Converter aisle at the Xstrata Nickel Smelter in Sudbury [46]

the number and dimensions of the converters varies. Thus it is important to develop concepts that are adequately transferable. This level of generality has eluded other researchers in the metallurgical industry [47], but is attained within this document.

CHAPTER 2

SEMI-DISCRETE DYNAMICS OF PS SYSTEMS

2.1 Gantt Structure

2.1.1 Assignments

A Peirce-Smith converting aisle is a semi-discrete dynamical system [48, 49]; it is characterized by a continuous evolution, except at fixed moments in time, when the system experiences discrete changes. These discrete changes are depicted in a Gantt chart (Figure 2.1) whenever an object begins or completes an assignment within the given schedule.

A converting aisle consists of several object classes, including converters, cranes, ladles, etc. $\mathcal{C}$ is the set of object classes that are included in the mathematical representation. Throughout this document, the converter class is denoted PSC. If cranes are the only other objects under consideration, then $\mathcal{C} = \{\text{PSC, Crane}\}$, for instance. Figure 2.1 considers three object classes, $\mathcal{C} = \{\text{PSC, Crane, OffgasTreatment}\}$.

The objects that must be coordinated in order to minimize delays and additional costs are said to be critical. If there are critical object classes that have been omitted from $\mathcal{C}$, then the mathematical representation is inadequate. Generally, the noncritical objects can be coordinated *a posteriori*, once a master schedule has been established for the critical objects.

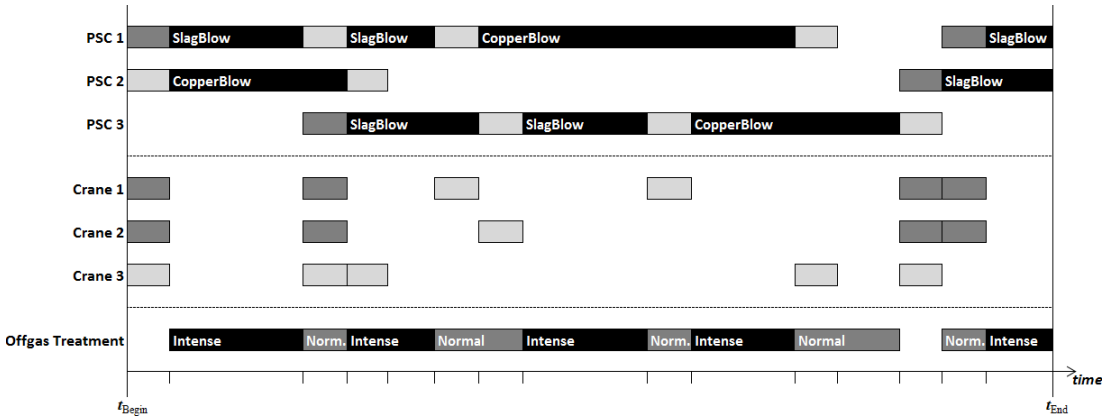


Figure 2.1: Gantt chart of a Peirce-Smith converting aisle. The schedule begins at time t_{Begin} and ends at time t_{End}. The discrete events are marked with short dashes along the time axis.

An individual object of some class $i \in \mathcal{C}$ is represented by an ordered pair (i, j), where j is a strictly positive integer. For example, (PSC,3) refers to Peirce-Smith Converter 3, and (Crane,1) refers to Crane 1, etc. Each of these objects is represented by a row in a Gantt chart (Figure 2.1).

The objects each undergo a sequence of assignments, which correspond to the sequence of coloured blocks within each Gantt row. An individual assignment is denoted by an ordered triple (i, j, k), such that (i, j) identifies the object, and k is the sequence number. For example, (PSC,3,2) refers to the second assignment of Peirce-Smith Converter 3.

The number of objects within class i that are to be included in the optimization is denoted n_i. Figure 2.1 depicts a system in which $n_{\text{PSC}} = n_{\text{Cranes}} = 3$, and $n_{\text{OffgasTreatment}} = 1$. Also, through practical considerations, there is always an upper bound $\overline{n}_{\text{Asgn}i}$ on the number of assignments that can be performed by the members of each object class within a given schedule. For example, if $\overline{n}_{\text{Asgn,PSC}} = 50$, then any schedule having more than 50 assignments for a converter is considered too complex to be feasible.

The set of assignments considered in the optimization is given by

$$\begin{aligned} \mathcal{A} \;=\; \{(l_1, l_2, l_3)|\;\; & l_1 \in \mathcal{C}, \\ & l_2 \in \{1, 2, \ldots, n_{l_1}\}, \\ & l_3 \in \{0, 1, 2, \ldots, \overline{n}_{\text{Asgn}l_1}\} \;\; \} \end{aligned}$$

Thus (l_1, l_2, l_3) identifies the $l_3{}^{\text{th}}$ assignment of the $l_2{}^{\text{th}}$ member of class l_1.

The value $l_3 = 0$ is used to identify the final assignments of the previous schedule, which may continue into the current schedule, or may end before the current schedule. The subset $\mathcal{A}_\circ = \{(l_1, l_2, l_3) \in \mathcal{A}|\; l_3 = 0\}$ is used to define the initial conditions of the current schedule.

The assignments belonging to a given object (i, j) form the subset $\mathcal{A}_{ij} = \{(l_1, l_2, l_3) \in \mathcal{A}|\; l_1 = i, l_2 = j\}$, which isolates a Gantt row. Similarly, the assignments belonging to an object class i form the subset $\mathcal{A}_i = \{(l_1, l_2, l_3) \in \mathcal{A}|\; l_1 = i\}$, which isolates the group of Gantt rows belonging to class i.

It is convenient to use a single index l to abbreviate the triple indices, as in $l = (l_1, l_2, l_3)$. This is complemented by the following two mappings.

$$\text{class} : (l_1, l_2, l_3) \mapsto l_1$$

is used to identify the object class of an assignment, and

$$\text{obj} : (l_1, l_2, l_3) \mapsto (l_1, l_2)$$

is used to identify the object of an assignment. Extending the previous notation, $\mathcal{A}_{\text{class}(l)}$ and $\mathcal{A}_{\text{obj}(l)}$ identify all of the assignments that are applied to the same class as l, and to the same object as l, respectively.

For a given assignment $l \in \mathcal{A}$, the immediate predecessor is denoted "$l-$". Formally, if $l = (l_1, l_2, l_3)$, then $l- = (l_1, l_2, l_3 - 1)$. (A similar notation "$l+$" can be adopted for the immediate successor, $l+ = (l_1, l_2, l_3 + 1)$, but this is not actually needed in the current formulation). The object mapping is such that $\text{obj}(l-) = \text{obj}(l)$, which reasserts that the assignment sequence is confined to a given object (Gantt row).

In addition to belonging to a given object, each assignment $l \in \mathcal{A}$ has the following properties,

- d^l is the planned duration of l.
- t^l is the planned completion time of l.
- Type^l is the planned assignment type of l.

These properties are depicted in Gantt charts, as d^l is the horizonal length of the assignment, and t^l can be read directly from the horizontal axis. The Type^l variables are used to categorize the assignment type, and may carry a qualitative description of l; the values of Type^l are often associated to the assignment colours and/or labels in a Gantt chart (Figure 2.1).

The durations d^l and the completion times t^l are continuous real variables, which are subject to linear inequalities. For example,

$$t^l - d^l \geq t^{l-} \tag{2.1}$$

compares the beginning time ($t^l - d^l$) of assignment l to the completion time t^{l-} of assignment $l-$. Indeed, l can only begin after its predecessor $l-$ has been completed. Such variables and inequalities are easily implemented using linear programming, the most elementary form of constrained optimization (Appendix B.1).

On the other hand, the assignment types Type^l are categorical variables [49, 50], rather than numerical variables. Categorical variables can be implemented into linear programs

only indirectly [50], through the use of binary variables, as in Chapter 4. The direct implementation of categorical variables will be discussed briefly in Section 6.1.

For each object class $i \in \mathcal{C}$, there is a set of assignment types denoted $\mathcal{T}_i$. For example, $\mathcal{T}_{\text{Crane}}$ = {AssistInitialCharge, AssistRecharge, AssistSkim, AssistFinalDischarge} lists four types of crane assignments. $\text{Type}^{(\text{Crane},j,k)}$ = AssistInitialCharge implies that assignment (Crane,j,k) is to assist the initial charge of a converter; likewise, $\text{Type}^{(\text{Crane},j,k)}$ = AssistRecharge implies that the assignment is to assist in the recharging of a converter, etc.

The domain of Type^l is given by $\mathcal{T}_{\text{class}(l)} \bigcup \{\text{Undetermined}\}$, for all $l \in \mathcal{A}$. The null category "Undetermined" applies to all classes. Type^l = Undetermined implies that l and all of its successors are to be planned in future schedules, beyond the current schedule.

If d^l, t^l and Type^l are known for all $\mathcal{A}$, then the corresponding Gantt chart can be constructed automatically. The values of d^l and t^l provide the outline, while the values of Type^l provide the colouring.

2.1.2 Dependencies

The set $\mathcal{A}$ describes the order of the assignments within each individual Gantt row. However, it does not describe the dependencies of assignments across several Gantt rows.

For example, suppose that an empty converter undergoes an InitialCharge assignment, which uses two cranes to carry a sequence of ladles to the converter. Thus an InitialCharge assignment is accompanied by two simultaneous crane assignments that are in support of the converter (Figure 2.2).

For each object class $i \in \mathcal{C}$, and each assignment type $k \in \mathcal{T}_i$, there is a set of dependency

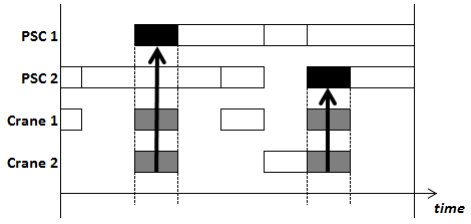


Figure 2.2: Depiction of assignment dependency. In this case, the charging of an empty converter (black) requires the assistance of two cranes (grey).

clauses $\mathcal{D}_{ik}$,

$$\begin{aligned}\mathcal{D}_{ik} \quad \subseteq \quad \{(i',k',n)| \quad & i' \in \mathcal{C}, \\ & k' \in \mathcal{T}_{i'} \text{ such that } (i',k') \neq (i,k), \\ & n \in \{1,2,\ldots,n_{i'}\} \qquad\qquad \}\end{aligned}$$

The set is defined such that $(i',k',n) \in \mathcal{D}_{ik}$ implies that whenever an assignment $l \in \mathcal{A}_i$ is of type $k \in \mathcal{T}_i$, there must be n assignments in $\mathcal{A}_{i'}$ of type $k' \in \mathcal{T}_{i'}$ that are performed simultaneously to l, in support of l. Two assignments $l, l' \in \mathcal{A}$ are simultaneous if $d^l = d^{l'}$ and $t^l = t^{l'}$.

The current formulation does not consider the case where $(i',k') = (i,k)$, as the logical meaning of $(i,k,n) \in \mathcal{D}_{ik}$ is unclear. One possible interpretation is that the type $k \in \mathcal{T}_i$ can only occur as groups of n simultaneous assignments of the same type. An equivalent effect can be obtained by creating a dummy class $i' \in \mathcal{C}$ with an assignment type $k' \in \mathcal{T}_{i'}$ such that $(i',k',1) \in \mathcal{D}_{ik}$; an additional assignment type $k'' \in \mathcal{T}_i$ must also be created such that $(i,k'',n-1) \in \mathcal{D}_{i'k'}$.

Returning to the InitialCharge example, a converter assignment $l \in \mathcal{A}_{\text{PSC}}$ of type InitialCharge $\in \mathcal{T}_{\text{PSC}}$ requires the support of two simultaneous crane assignments of type InitialCharge $\in \mathcal{T}_{\text{Crane}}$. It follows that (Crane, InitialCharge, 2)$\in \mathcal{D}_{\text{PSC,InitialCharge}}$.

Ultimately, the simultaneous coordination of objects is the constrained optimization of d^l, t^l and Type^l for all $l \in \mathcal{A}$. The Peirce-Smith Converter Problem is distinguished from a general coordination problem because of the particularities discussed in Section 1.3, which require a more detailed treatment of the PSC class.

2.2 PS Converters as State-Machines

2.2.1 States and Transitions

An object may be considered a state-machine [51] if it is described by (1) so-called state variables that describe the state of the object at different times, and (2) transitions that mark the machine-like departures from one state to the next.

To embed the structure of a state-machine within the general Gantt structure, transitions are interpreted as assignments. Thus if $i \in \mathcal{C}$ is a state-machine class, then $\mathcal{A}_{ij}$ is the set of transitions (assignments) that are applied to the state-machine (i,j). As before, transition l is categorized by Type^l.

Within the Peirce-Smith Converter Problem, the converters are implemented as state-

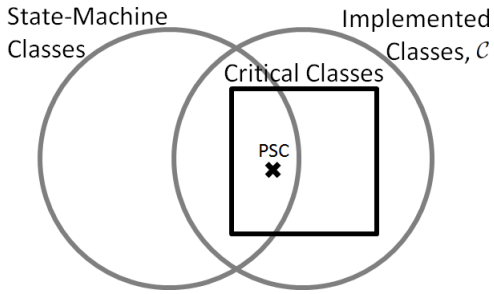


Figure 2.3: Venn diagram describing the object classes of a Peirce-Smith system. The PSC class is a critical state-machine class.

machines (Figure 2.3). Depending on the context, it may be necessary to implement other object classes as state-machines, following the example set forth in this document for the PSC class. This topic will be revisited in Section 6.2.

In the current implementation of the PSC class, the state description has three components,

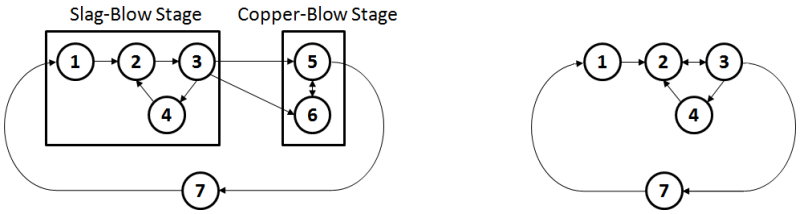
- The mass content of the streams that are retained in the bath of a converter
- The heat content that is retained in the bath of a converter
- The mechanistic aspects of the converter state

Variables describing the retained mass and heat content are developed in the following two chapters. As an example, $m^l_{\text{FeS,RetProd}}$ is the mass of FeS that is retained in the product streams of obj(l) at time t^l. Similarly, h^l_{Ret} is the relative heat that is retained in the bath of obj(l) at time t^l.

The mechanistic aspects of a converter state are described by the Type^l variables. Depending on the previous transition type, a converter may or may not be mechanistically prepared for following transition type. Thus the Type^l variables have a double-role, categorizing states as well as transitions.

This notion of mechanistic preparedness is made more precise by imposing a structure on $\mathcal{T}_{\text{PSC}}$. Firstly, the subset $\mathcal{T}_{\text{PSC,Empty}} \subset \mathcal{T}_{\text{PSC}}$ contains all of the transition types for which the converter is left empty; an empty converter does not contain a bath, hence cannot retain any mass or heat. Additionally, for every $k \in \mathcal{T}_{\text{PSC}}$, there is a set of preceding transition types $\mathcal{T}^-_{\text{PSC}k} \subset \mathcal{T}_{\text{PSC}}$, such that

$$(\text{Type}^l = k) \text{ implies } (\text{Type}^{l-} \in \mathcal{T}^-_{\text{PSC}k})$$



(a) (b)

Figure 2.4: Converter transition diagrams for (a) typical copper PS systems, and (b) typical nickel PS systems. The transition types are numbered (1) InitialCharge, (2) SlagBlow, (3) Skim, (4) Recharge, (5) CopperBlow, (6) ScrapCharge, and (7) EndCycle.

Hence a converter is mechanistically prepared for a transition of type $k \in \mathcal{T}_{\text{PSC}}$ if the preceding transition was of a type $k' \in \mathcal{T}^{-}_{\text{PSC}k}$.

Figure 2.4a describes a typical Peirce-Smith system for copper production, in which

$$\mathcal{T}_{\text{PSC}} = \{\text{InitialCharge}, \text{SlagBlow}, \text{Skim}, \text{Recharge}, \text{CopperBlow}, \text{ScrapCharge}, \text{EndCycle}\}$$

Figure 2.4b represents a typical nickel producing system, which does not include a Copper-Blow Stage.

In a copper PS system (Figure 2.4a), a converter may be in a state represented by node 3; this means that the preceding transition was a Skim. According to the diagram, the converter is therefore mechanistically prepared for a Copper-Blow, since

$$\mathcal{T}^{-}_{\text{PSC, CopperBlow}} = \{\text{Skim}, \text{ScrapCharge}\}$$

includes Skim. Thus, if Type^{l-}=Skim, then the converter is mechanistically prepared for Type^{l}=CopperBlow, but this is not a sufficient condition to allow a Copper-Blow; indeed, if $m^{l-}_{\text{FeS,RetProd}} \neq 0$, then there is iron in the system, and the Copper-Blow is not thermodynamically feasible (See Subsection 1.3.2).

In general, a transition of a certain type can be performed only if the state satisfies mechanistic constraints, as well as heat and mass constraints. These considerations are formalized within the MILP of Chapter 4.

2.2.2 Converting Actions

Converter dynamics can be decomposed into a sequence of transitions, which in turn can be decomposed into a sequence of actions. Even though the transitions vary in type, they generally contain a common set of converting actions: charge, blow and discharge.

The set of converting actions is denoted

$$\{\text{Ch}, \text{Blow}, \text{DCh}\}$$

where "Ch" and "DCh" are short for "Charge" and "Discharge", respectively. In this context, skimming is considered to be a kind of discharging.

Before running the optimization, there is no direct way of knowing which type of transitions or actions will be performed, and in what sequence. For example, (PSC,2,3) is the third transition of converter 2; under optimality, this transition might include a blowing action, or it might not. Every converting action variable must be defined for (PSC,2,3), otherwise the MILP formulation is not free to evaluate which actions to include or exclude in this transition, e.g. whether or not (PSC,2,3) should include blowing.

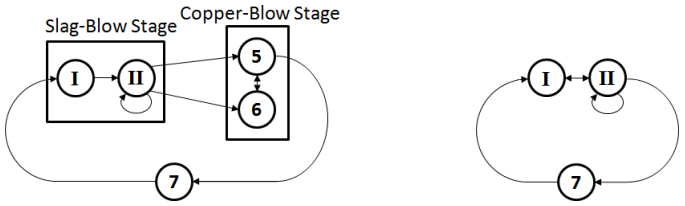
For simplicity, the converter transitions will take the following generic form,

$$\text{Ch} - \text{Blow} - \text{DCh}$$

Given that each action is present once, this ordering tends to minimize the number of converting transitions needed to construct the schedule; it allows charging to be followed by blowing and/or discharging, or blowing to be followed by discharging, within the same transition.

Not all converting actions are included in all transition types. For example, there may be some transition types that allow charging and blowing, but do not allow any discharging, thus giving sequences of the form $\text{Ch} - \text{Blow}$. The generic action sequence is still respected, as long as (1) converting actions are not repeated within the same transition, and (2) the generic order is respected.

In Figure 2.4, most of the transition types consider a single converting action, which can be deduced for the type name. It is clear, for example, that InitialCharge consists of the Ch action, and does not include Blow or DCh. However, the EndCycle could well include an overblow prior to the final discharge, hence the sequence Blow-DCh, which still agrees with the generic form.



(a) (b)

Figure 2.5: Converter transition diagrams for (a) Simplified copper PS systems, and (b) Simplified nickel PS System. The transition types are labeled (I) BeginSlagBlowStage, (II) ContinueSlageBlowStage, and the remainder are numbered as in Figure 2.4.

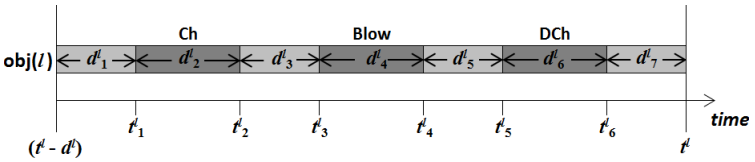


Figure 2.6: Generic converter transition

Considering that InitialCharge transitions are always followed by SlagBlow and Skim (Figure 2.4), the InitialCharge type may be replaced with a new type of transition, say BeginSlagBlowStage, still satisfying the generic form. Similarly the Recharge transition can be replaced by a new transition type, ContinueSlageBlowStage, consisting of the original Recharge, followed by SlagBlow and Skim (Figure 2.5). These types of mergers decrease the size of $\mathcal{T}_{\text{PSC}}$, thus simplifying the parametrization, and shortening the computation time.

However, some transition types must not be merged, because it would effect the dependency clauses $\mathcal{D}_{\text{PSC}k}$. For example, the dependency (Crane,InitialCharge,2)$\in \mathcal{D}_{\text{PSC,InitialCharge}}$, discussed in Section 2.1.2, could not be articulated if InitialCharge were merged into BeginSlagBlowStage. In general, it is advisable to merge transitions as much as possible, while maintaining the relevant dependency clauses.

Figure 2.6 illustrates a transition over the timeline, in accordance with the generic form. The transition duration d^l is decomposed into seven time segments; this includes the three converting actions, as well as intermittent setup and/or idle times. Thus,

$$d^l = \sum_{i=1}^{7} d_i^l \tag{2.2}$$

The even terms correspond to the converting actions, such that $d_2^l = d_{\text{Ch}}^l$ is the charging

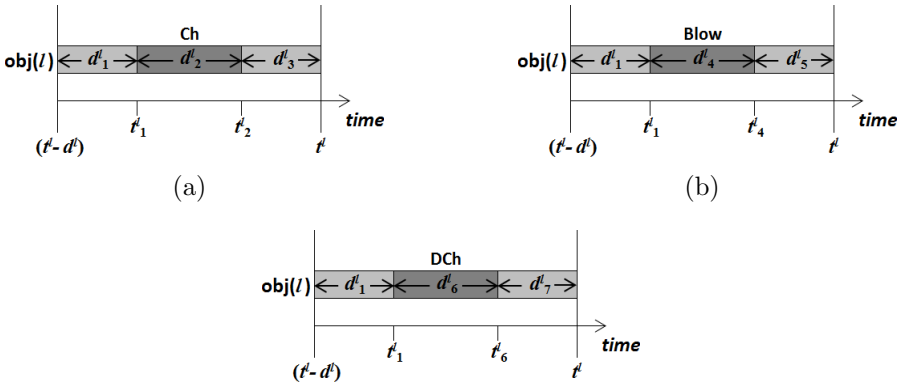


Figure 2.7: Converter transitions having one converting action

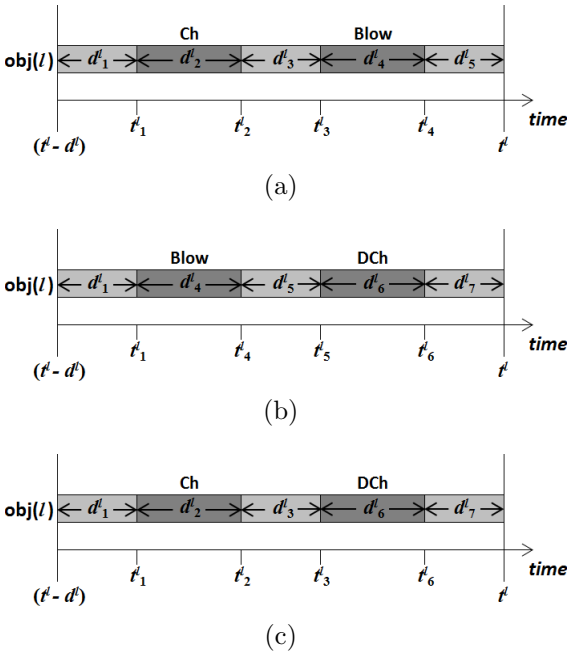


Figure 2.8: Converter transitions having two converting actions

duration, $d_4^l = d_{\text{Blow}}^l$ is the blowing duration, and $d_6^l = d_{\text{DCh}}^l$ is the discharging duration. Transitions which do not include charging are constrained such that $d_{\text{Ch}}^l = 0$; similarly, $d_{\text{Blow}}^l = 0$ when there is no blowing, and $d_{\text{DCh}}^l = 0$ when there is no discharging.

Figures 2.7 and 2.8 illustrate transitions having one and two converting actions, respectively. The Ch-DCh form (Figure 2.8c) has been included for completeness, even though it is not practical; proceeding from a Ch into DCh, without the essential mixing that occurs during the blowing action [7], would yield heterogeneous product streams. Figures 2.7 and 2.8 illustrate the convention regarding the setup/idle times: to keep the lowest indices possible, while respecting $d_2^l = d_{\text{Ch}}^l$, $d_4^l = d_{\text{Blow}}^l$, and $d_6^l = d_{\text{DCh}}^l$. Purely mechanistic transitions are such that $d^l = d_1^l$.

The generic form provides a unified framework to construct transition types. Under this framework, transition types are distinguished through the various process streams that are involved in the charging/blowing/discharging actions, which is the subject of Chapter 3.

CHAPTER 3

CHARACTERIZATION OF CHEMICAL STREAMS IN PS SYSTEMS

3.1 Elements, Species and Streams

3.1.1 General Representation of Chemical Converting

In Peirce-Smith converting, and in many other chemical processes, feed materials are loaded into the system and are reacted to form certain chemical species; these species are eventually separated into the product streams (Figure 3.1). Thus the formation of product species is preliminary to the formation of the product streams.

The set of chemical elements which participate in the process is denoted $\mathcal{E}$. These chemical elements are organized into species which, in turn, are organized into streams. The set of chemical species and streams are denoted $\mathcal{S}$ and $\mathcal{Z}$, respectively.

To facilitate the discussion of elements, species and streams, a certain convention is used to index the sets $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{Z}$. Elements are indexed using the subscript i, e.g. m_i is a mass quantity of element $i \in \mathcal{E}$. Secondly, species are indexed using the subscript j, e.g. m_j is a mass quantity of species $j \in \mathcal{S}$. Thirdly, process streams are indexed using the subscript k, e.g. m_k is a mass quantity of stream $k \in \mathcal{Z}$. The alphabetical ordering ijk is logical, since elements are the components of species, which are then the components of process streams.

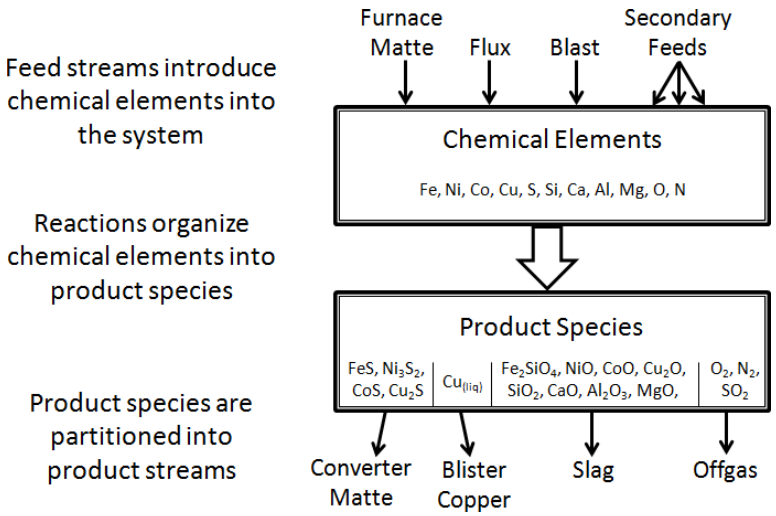


Figure 3.1: Elemental mass distribution, for the General Nickel-Copper PSC Formulation

The set of streams $\mathcal{Z}$ includes feeds $\mathcal{Z}_{\text{Feed}}$ which are introduced into the converters, and products $\mathcal{Z}_{\text{Prod}}$ which are removed or expelled from the converters. Thus $\mathcal{Z} = \mathcal{Z}_{\text{Feed}} \bigcup \mathcal{Z}_{\text{Prod}}$, and $\mathcal{Z}_{\text{Feed}} \bigcap \mathcal{Z}_{\text{Prod}} = \varnothing$.

For every stream $k \in \mathcal{Z}$, there is a set of constituent species $\mathcal{S}_k \subset \mathcal{S}$ and a set of constituent elements $\mathcal{E}_k \subset \mathcal{E}$. Thus, the sets of feed species and product species are given by

$$\mathcal{S}_{\text{Feed}} = \bigcup_{k \in \mathcal{Z}_{\text{Feed}}} \mathcal{S}_k$$

$$\mathcal{S}_{\text{Prod}} = \bigcup_{k \in \mathcal{Z}_{\text{Prod}}} \mathcal{S}_k$$

Similarly for the feed and product elements,

$$\mathcal{E}_{\text{Feed}} = \bigcup_{k \in \mathcal{Z}_{\text{Feed}}} \mathcal{E}_k$$

$$\mathcal{E}_{\text{Prod}} = \bigcup_{k \in \mathcal{Z}_{\text{Prod}}} \mathcal{E}_k$$

The chemical reactions that occur during the converting process cause a reorganization of the species, so that $\mathcal{S}_{\text{Feed}}$ is not generally equal to $\mathcal{S}_{\text{Prod}}$. In contrast, it can be argued that $\mathcal{E}_{\text{Feed}} = \mathcal{E}_{\text{Prod}} = \mathcal{E}$, since every element is introduced as part of the feeds, and reports to a least one of the products.

Figure 3.1 summarizes the logic by which feeds $\mathcal{Z}_{\text{Feed}}$ are entered into the system, carrying elements $\mathcal{E}$ that are subsequently organized into product species $\mathcal{S}_{\text{Prod}}$, from which the product streams $\mathcal{Z}_{\text{Prod}}$ are drawn. This is the logic by which elemental masses are distributed and organized within the system, $\mathcal{Z}_{\text{Feed}} \rightarrow \mathcal{E} \rightarrow \mathcal{S}_{\text{Prod}} \rightarrow \mathcal{Z}_{\text{Prod}}$, which does not rely on $\mathcal{S}_{\text{Feed}}$.

Knowledge of the feed species $\mathcal{S}_{\text{Feed}}$ can be quite limited, particularly for secondary feeds. Although the balancing of elemental mass is well-posed even without this knowledge, the fundamental predictions for volume and temperature can suffer. Nonetheless, empirical measurements and plant studies can compensate for this lack of information, as discussed in Subsection 3.3.2.

The structures represented by $\mathcal{E}$, $\mathcal{S}_{\text{Feed}}$, $\mathcal{S}_{\text{Prod}}$, $\mathcal{Z}_{\text{Feed}}$ and $\mathcal{Z}_{\text{Prod}}$ are general to any chemical conversion process, including solvent extraction, steelmaking, etc. The remainder of the chapter focusses on the particularities of Peirce-Smith converting.

3.1.2 General Representation of PS Converting

A general treatment of the Peirce-Smith converting requires an appropriate consideration of the elements, species and streams that are involved. Thus the General Nickel-Copper PSC Formulation is presented along side the Simplified Copper PSC Formulation, sharing the same algebraic structures. The former is intended to represent the conversion of either nickel-copper matte or of copper matte. The Simplified Copper PSC Formulation is dedicated to the latter.

PS feeds generally include furnace matte, flux, various types of secondary feeds (reverts, scraps, etc.) and a blend of blast streams. Thus,

$$\mathcal{Z}_{\text{Feed}} \;=\; \{\;\; \text{FMatte, Flux, } \ldots \;\; \} \;\bigcup\; \mathcal{Z}_{\text{Blast}}$$

The prefix "F" in FMatte distinguishes the furnace matte (or feed matte) from CMatte, which is the Bessemer converter matte that is an element of $\mathcal{Z}_{\text{Prod}}$.

As discussed in Subsection 1.3.1, the blast streams $\mathcal{Z}_{\text{Blast}}$ are characterized by particular parameters including the oxygen enrichment and the blast rate. Usually the blast consists of a regular air stream, plus a stream that is highly enriched with oxygen [7]; the net enrichment of the blast is controlled by blending these two streams. Given the particularity of the blast, a distinction is made between $\mathcal{Z}_{\text{Blast}}$ and the remaining feeds that are nongaseous. $\mathcal{Z}_{\text{NGFeed}} = \mathcal{Z}_{\text{Feed}} \setminus \mathcal{Z}_{\text{Blast}}$ is used to denote the nongaseous feeds.

Depending on the operation, there may be a second stream of furnace matte, denoted FMatte2 $\in \mathcal{Z}_{\text{NGFeed}}$; for example, the Chiquicamata Smelter operates a flash furnace in parallel with a reverberatory furnace [47], and the two furnaces each produce different mattes. There may also be several additional blends of flux, $\{\text{Flux2, Flux3, } \ldots\} \subset \mathcal{Z}_{\text{NGFeed}}$. Furthermore, there may be several secondary feeds, e.g. $\{\text{Scrap, Scrap2, Reverts}\} \subset \mathcal{Z}_{\text{NGFeed}}$. Generally, $\mathcal{Z}_{\text{NGFeed}}$ is constructed on a smelter-to-smelter basis.

In the General Nickel-Copper PSC Formulation, the feed streams are presumed to carry the following elements,

$$\mathcal{E} \;=\; \{\;\; \text{Fe, Ni, Co, Cu, S, Si, Ca, Al, Mg, O, N} \;\; \}$$

which are organized into the following set of product species,

$$\mathcal{S}_{\text{Prod}} = \{ \begin{array}{l} \text{FeS}_{(\text{liq})},\ \text{Ni}_3\text{S}_{2(\text{liq})},\ \text{CoS}_{(\text{liq})},\ \text{Cu}_2\text{S}_{(\text{liq})},\ \text{Cu}_{(\text{liq})},\ \text{Fe}_2\text{SiO}_{4(\text{liq})},\ \text{Fe}_3\text{O}_{4(\text{sol})},\ \text{NiO}_{(\text{sol})}, \\ \text{CoO}_{(\text{sol})},\ \text{Cu}_2\text{O}_{(\text{sol})},\ \text{SiO}_{2(\text{sol})},\ \text{CaO}_{(\text{sol})},\ \text{Al}_2\text{O}_{3(\text{sol})},\ \text{MgO}_{(\text{sol})},\ \text{O}_{2(\text{gas})},\ \text{N}_{2(\text{gas})},\ \text{SO}_{2(\text{gas})} \end{array} \}$$

The Simplified Copper PSC Formulation disregards the nickel and cobalt, thereby reducing $|\mathcal{E}|$ from 11 to 9, and $|\mathcal{S}_{\text{Prod}}|$ from 17 to 13.

Each species is defined by a stoichiometric expression, e.g. FeS, and a state-of-matter, be it solid (sol), liquid (liq) or gas. It is usually convenient to drop the state-of-matter, except for $\text{Cu}_{(\text{liq})}$; in this case the subscript distinguishes the liquid metallic species $\text{Cu}_{(\text{liq})} \in \mathcal{S}_{\text{Prod}}$ from the element $\text{Cu} \in \mathcal{E}$. In the present context, whenever a species $j \in \mathcal{S}_{\text{Prod}}$ is given simply as a stoichiometric expression, the state-of-matter can be inferred; FeS implies $\text{FeS}_{(\text{liq})}$, SiO_2 implies $\text{SiO}_{2(\text{sol})}$, etc.

As depicted in Figure 3.1, the set of product streams is given by

$$\mathcal{Z}_{\text{Prod}} = \{ \text{ CMatte, Blister, Slag, Offgas } \}$$

Nickel producing systems do not usually extract blister copper from nickel-copper mattes because this results in massive nickel and cobalt losses to the slag; thus it may seem appropriate to omit Blister from $\mathcal{Z}_{\text{Prod}}$. However, the Blister stream is included for the sake of generality, describing the spectrum of nickel, nickel-copper and copper systems. After all, it is theoretically possible for a nickel smelter to produce blister copper (Subsection 1.3.2).

The nongaseous products are given by $\mathcal{Z}_{\text{NGProd}} = \mathcal{Z}_{\text{Prod}} \setminus \{\text{Offgas}\}$. Moreover, $\mathcal{Z}_{\text{NG}} = \mathcal{Z}_{\text{NGFeed}} \bigcup \mathcal{Z}_{\text{NGProd}}$ is the set of nongaseous streams.

In the General Nickel-Copper PSC Formulation, furnace mattes are represented reasonably well with

$$\mathcal{S}_{\text{FMatte}} = \{ \text{ FeS, } \text{Ni}_3\text{S}_2\text{, CoS, } \text{Cu}_2\text{S } \}$$

and in the Simplified Copper PSC Formulation with

$$\mathcal{S}_{\text{FMatte}} = \{ \text{ FeS, } \text{Cu}_2\text{S } \}$$

although there are more complicated formulations that include oxides and other species [52, 53]. The fluxing species vary from smelter to smelter, but are generally a blend of the stable oxides,

$$\mathcal{S}_{\text{Flux}} \subseteq \{ \text{ } \text{SiO}_2\text{, CaO, } \text{Al}_2\text{O}_3\text{, MgO } \}$$

The blast has a two-element (two-species) configuration,

$$\mathcal{S}_{\text{Blast}} \;=\; \{\;\; \text{O}_2,\, \text{N}_2 \;\;\}$$

where $\mathcal{S}_{\text{Blast}}$ is equivalent to $\bigcup_{k\in\mathcal{Z}_{\text{Blast}}} \mathcal{S}_k$. For the secondary feeds, the contributing species $\mathcal{S}_k$ are often unavailable, as discussed in Subsection 3.3.2.

However, the contributing species $\mathcal{S}_k$ are generally available for all of the product streams $k \in \mathcal{Z}_{\text{Prod}}$. In the General Nickel-Copper PSC Formulation,

$$\mathcal{S}_{\text{CMatte}} \;=\; \{\;\; \text{FeS},\, \text{Ni}_3\text{S}_2,\, \text{CoS},\, \text{Cu}_2\text{S} \;\;\}$$

$$\mathcal{S}_{\text{Blister}} \;=\; \{\;\; \text{Cu}_{(\text{liq})} \;\;\}$$

$$\mathcal{S}_{\text{Slag}} \;=\; \{\;\; \text{Fe}_2\text{SiO}_4,\, \text{Fe}_3\text{O}_4,\, \text{NiO},\, \text{CoO},\, \text{Cu}_2\text{O},\, \text{SiO}_2,\, \text{CaO},\, \text{Al}_2\text{O}_3,\, \text{MgO} \;\;\}$$

$$\mathcal{S}_{\text{Offgas}} \;=\; \{\;\; \text{O}_2,\, \text{N}_2,\, \text{SO}_2 \;\;\}$$

In the Simplified Copper PSC Formulation, the nickel and cobalt species are removed from $\mathcal{S}_{\text{Matte}}$ and $\mathcal{S}_{\text{Slag}}$. The blister is regarded as pure copper in spite of the $\approx 1\%$ impurities; these impurities are more relevant in the refining stages. The gaseous product species all report to the offgas stream, and the remaining nongaseous species form a set $\mathcal{S}_{\text{NGProd}} = \mathcal{S}_{\text{Prod}} \setminus \mathcal{S}_{\text{Offgas}}$.

In the remainder of Chapter 3, the General Nickel-Copper PSC Formulation and the Simplified Copper PSC Formulation are presented in tandem, as the differences are only superficial. Chapter 4 presents an MILP formulation that makes absolutely no distinction, which is truly unified approach.

3.2 Species-Based Distribution of Mass, Volume and Heat

3.2.1 Mass Distribution Within a Process Stream

The Peirce-Smith reactions are controlled through the chemical balance of feed and product streams. It is therefore important to consider the composition of the various streams that participate in the process.

The mass m_k of a stream k can be decomposed in terms of elemental contributions,

$$m_k = \sum_{i\in\mathcal{E}_k} m_{ik} \tag{3.1}$$

or in terms of species contributions,

$$m_k = \sum_{j \in \mathcal{S}_k} m_{jk} \tag{3.2}$$

These two decompositions are related to each other through the element-species mass balance for stream k,

$$m_{ik} = \sum_{j \in \mathcal{S}_k} w_{ij} m_{jk} \tag{3.3}$$

for all $i \in \mathcal{E}_k$, in which w_{ij} is the mass-fraction of element i that is contained in species j. These mass-fractions w_{ij} are universal constants that can be obtained using the stoichiometric descriptions of the species, as described below. Equation 3.3 describes how element i is distributed among the species of stream k.

The stoichiometric description of a species is essentially a set of molar proportions. For instance, the expression FeS implies that there is one mole of Fe and one mole of S, for every mole of FeS; similarly, Cu_2S implies that there are two moles of Cu and one of S for every mole of Cu_2S. These stoichiometric (molar) ratios are fundamental for the description of chemical reactions, as in Subsection 1.3.2. Feed and product tonnages, however, are based on mass ratios w_{ij} rather than molar ratios.

The molar ratios are hence converted into mass ratios, according to

$$w_{ij} = \frac{n_{ij} M_i}{M_j} \tag{3.4}$$

where n_{ij} is the number of moles of i contained in a mole of j, M_i is the molar weight of element i and M_j is the molar weight of species j. For an element i, M_i is taken directly from the periodic table as the standard atomic weight of i. For a compound j, M_j must be decomposed into the sum of its constituent atomic weights. For example, the mass-fraction of Cu in Cu_2S is given by

$$w_{\mathrm{Cu,Cu_2S}} = \frac{2M_{\mathrm{Cu}}}{M_{\mathrm{Cu_2S}}} = \frac{2M_{\mathrm{Cu}}}{2M_{\mathrm{Cu}} + M_{\mathrm{S}}} = \frac{2(63.546)}{2(63.546) + 32.065} = 0.79853$$

The factor 2 comes from the fact that there are two moles of Cu within a mole of Cu_2S, i.e. $n_{\mathrm{Cu,Cu_2S}} = 2$. Liquid copper $\mathrm{Cu_{(liq)}}$ is a particular case for which $w_{\mathrm{Cu,Cu_{(liq)}}} = 1$, and $w_{i\mathrm{Cu_{(liq)}}} = 0$ for all $i \neq$ Cu.

For a stream k, the elemental composition is given by

$$w_{ik} = \frac{m_{ik}}{m_k}$$

for all $i \in \mathcal{E}_k$, and the species composition is given by

$$w_{jk} = \frac{m_{jk}}{m_k}$$

for all $j \in \mathcal{S}_k$. Substitution into Equation 3.3 gives

$$w_{ik} = \sum_{j \in \mathcal{S}_k} w_{ij} w_{jk} \tag{3.5}$$

for all $i \in \mathcal{E}_k$. Thus the elemental composition $\{w_{ik} |\ i \in \mathcal{E}_k\}$ is a weighted mean of the species composition $\{w_{jk} |\ j \in \mathcal{S}_k\}$, in which the weights w_{ij} are known through Equation 3.4.

Using Equation 3.5, it is trivial to obtain the element composition from the species composition. However, the inverse problem of solving for $\{w_{jk} |\ j \in \mathcal{S}_k\}$ given $\{w_{ik} |\ i \in \mathcal{E}_k\}$, is called the "speciation of k", and it is not always trivial because the system of equations can be under-specified. In other words, there may be more candidate species in $\mathcal{S}_k$ than there are elements in $\mathcal{E}_k$.

Speciation is an important step in quantifying the volumetric and thermal contributions of the various feed streams. A speciation technique is discussed in Subsection 3.3.2, which is based on the geometrical notion of Convex Projection [54, 55], and is applicable to secondary feeds. More general techniques rely on thermodynamic and kinetic considerations (see [56] and [57], for example), and are beyond the current MILP implementation. Some of these techniques are implemented as software packages, e.g. FactSage©, ThermoCalc© and HSC©.

3.2.2 Volume Distribution Within a Process Stream

Peirce-Smith converters are of a fixed size. Therefore it is important to estimate the volume of the various streams that enter and exit the process.

The volume v_k of stream k includes contributions from the constituent species, giving the following approximate relationship,

$$v_k = \sum_{j \in \mathcal{S}_k} \frac{m_{jk}}{\rho_j} \tag{3.6}$$

where ρ_j is the density of species j, for all $j \in \mathcal{S}_k$.

The species densities ρ_j are given in Table 3.1, for all of the members of $\mathcal{S}_{\text{Prod}}$ except for the gaseous species $\{O_2, N_2, SO_2\}$. The densities of these gaseous species are given by

$$\rho_j = \frac{M_j P}{RT} \tag{3.7}$$

which is based on the Ideal Gas Law [58], in which P is the pressure, T is the temperature, and $R = 8.3145$ J/(mol °C) is the Ideal Gas Constant. Equation 3.7 assumes that T is provided in absolute terms, measured in K rather than °C.

The density of stream k is given by

$$\rho_k = \frac{m_k}{v_k}$$

According to Equation 3.6, ρ_k can therefore be estimated as

$$\rho_k = \left(\sum_{j \in \mathcal{S}_k} \frac{w_{jk}}{\rho_j} \right)^{-1} \tag{3.8}$$

Thus ρ_k is taken as the weighted harmonic mean of the constituent densities ρ_j.

There may be an upper limit $\overline{v}_k$ on the volume of stream k, such that $v_k \leq \overline{v}_k$. If the species composition of a stream can be estimated, then Equations 3.6-8 predict whether or not v_k will exceed this limit.

3.2.3 Heat Distribution Within a Process Stream

Heat and temperature control is a persistent concern in pyrometallurgical operations. It is therefore important in the PSC Problem to estimate the heat content of the various streams. In turn, the heat content is integrally related to the temperature.

The heat content h_k of a stream k can be expressed as

$$h_k(T) = \sum_{j \in \mathcal{S}_k} [w_{\text{H}j}(T)] m_{jk} \tag{3.9}$$

where $w_{\text{H}j}$ is the specific heat content of species j, which is a function of the equilibrium temperature T, described below. The notation $w_{\text{H}j}$ emphasizes a certain analogy to the elemental mass-fractions w_{ij}. In some sense, the enthalpy H can be regarded as if it were an

Table 3.1: Density and temperature response parameters for product species in Peirce-Smith systems [59, 60]

j	ρ_j [g/L]	$w_{\mathrm{H}j}^{\mathrm{Ref}}$ [J/g]	A_j [J/(g°C)]	B_j [J/(g(°C)^2)]	C_j [(J°C)/g]	D_j [J/(g(°C)^3)]
$FeS_{(liq)}$	5.27	-735.18	0.71153	0	0	0
$Ni_3S_{2(liq)}$	5.17	-0.70833	0.79845	0	0	0
$CoS_{(liq)}$	5.45	0.32968	0.76925	0	0	0
$Cu_2S_{(liq)}$	5.28	-427.87	0.56336	0	0	0
$Cu_{(liq)}$	7.92	204.86	0.49403	0	0	0
$Fe_2SiO_{4(liq)}$	2.5	-7155.9	1.1806	0	0	0
$Fe_3O_{4(sol)}$	5.2	-4825.2	0.74430	$3.4024{\cdot}10^{-4}$	-17708.	0
$NiO_{(sol)}$	7.45	-3272.7	0.63325	$1.2049{\cdot}10^{-4}$	0	0
$CoO_{(sol)}$	5.68	-3212.0	0.68500	$3.6299{\cdot}10^{-5}$	37127.	$5.7852{\cdot}10^{-8}$
$Cu_2O_{(sol)}$	6.0	-1257.9	0.43567	$1.6668{\cdot}10^{-4}$	0	0
$SiO_{2(sol)}$	2.65	-14166.	0.75726	$6.0693{\cdot}10^{-4}$	-16803.	0
$CaO_{(sol)}$	3.32	-11323.	0.74643	$3.6127{\cdot}10^{-4}$	-8061.5	0
$Al_2O_{3(sol)}$	3.99	-16384.	0.90645	$3.6829{\cdot}10^{-4}$	-21450.	0
$MgO_{(sol)}$	3.65	-14938.	1.1279	$1.2431{\cdot}10^{-4}$	-21674.	0
$O_{2(gas)}$	(Ideal Gas)	0	1.0818	$3.3749{\cdot}10^{-5}$	-24553.	0
$N_{2(gas)}$	(Ideal Gas)	0	0.97124	$1.4942{\cdot}10^{-4}$	0	0
$SO_{2(gas)}$	(Ideal Gas)	-4635.1	0.50310	$3.4629{\cdot}10^{-4}$	0	$-5.4231{\cdot}10^{-8}$

element, and it is distributed among species in a manner comparable to Equation 3.3.

A complete description of the temperature distribution would require the solution of partial differential equations over space and time [61]. However, Peirce-Smith converting has favourable mixing characteristics [7], so that the spacial (geometrical) aspects can be neglected for the current study. Thus the equilibrium temperature T can be regarded as a uniform temperature that is held throughout the stream.

A more precise description of T follows: if the stream were to be isolated indefinitely from other bodies (i.e. if it were kept under adiabatic conditions, [62]), then regardless of the initial temperature distribution, the heat h_k would rearrange itself among the constituent masses m_{jk}, so that the temperature distribution would tend toward one uniform value, T. Moreover, this temperature T may result from an equilibrium between several streams that are in thermal contact, as discussed in Subsection 3.2.4.

The specific heat content $w_{\mathrm{H}j}$ is a quantity of heat per unit mass, measured in J/g or equivalently in MJ/T (i.e. mega-joules per metric tonne). This is not to be confused with the specific heat capacity c_j of species j, which is the partial derivative of $w_{\mathrm{H}j}$ with respect

to temperature. The specific heat capacity is measured in J/(g·K), in MJ/(T·K), in J/(g·°C) or in MJ/(T·°C), which are all equivalent.

From the Fundamental Theorem of Calculus [63],

$$w_{\mathrm{H}j}(T) = w_{\mathrm{H}j}^{\mathrm{Ref}} + \int_{T^{\mathrm{Ref}}}^{T} c_j(T')dT'$$

where $w_{\mathrm{H}j}^{\mathrm{Ref}}$ is the specific heat of formation of species j at the reference temperature T^{Ref}, assuming that the pressure is held at a constant reference pressure P^{Ref}. The specific heat capacity is a continuous function of T, so that $w_{\mathrm{H}j}(T^{\mathrm{Ref}}) = w_{\mathrm{H}j}^{\mathrm{Ref}}$. By standard convention, T^{Ref} is taken as 298.15 K (25.00°C), P^{Ref} is taken to be atmospheric pressure 101.325 kPa, and $w_{\mathrm{H}j}^{\mathrm{Ref}}$ is taken such that $w_{\mathrm{H}j}^{\mathrm{Ref}} = 0$ if j is the most stable form of some pure element i under standard conditions [62]; for example, $w_{\mathrm{H,O_2}}^{\mathrm{Ref}} = 0$ because $\mathrm{O_2}$ gas is the most stable species of pure oxygen at 25°C and atmospheric pressure. Table 3.1 includes values of $w_{\mathrm{H}j}^{\mathrm{Ref}}$ for all species in $\mathcal{S}_{\mathrm{Prod}}$.

Within the table, $w_{\mathrm{H,Cu_{(liq)}}}^{\mathrm{Ref}}$ is taken to be the standard heat of fusion of copper [59], noting that solid copper $\mathrm{Cu_{(sol)}}$ satisfies $w_{\mathrm{H,Cu_{(sol)}}}^{\mathrm{Ref}} = 0$, as copper is naturally a solid at room temperature and atmospheric pressure.

The table also includes coefficients (A_j, B_j, C_j, D_j) which can be used to estimate c_j over a sufficiently wide range of temperature values,

$$c_j(T) = A_j + B_j T + C_j T^{-2} + D_j T^2 \tag{3.10}$$

Collectively, $w_{\mathrm{H}j}^{\mathrm{Ref}}$ and (A_j, B_j, C_j, D_j) are the temperature response parameters for species j, as they describe the temperature response of j to a given amount of heat. Combining Equation 3.10 and the Fundamental Theorem of Calculus, it follows that

$$w_{\mathrm{H}j}(T) = w_{\mathrm{H}j}^{\mathrm{Ref}} + A_j(T - T^{\mathrm{Ref}}) + \frac{B_j}{2}\left(T^2 - (T^{\mathrm{Ref}})^2\right) - C_j\left(T^{-1} - (T^{\mathrm{Ref}})^{-1}\right) + \frac{D_j}{3}\left(T^3 - (T^{\mathrm{Ref}})^3\right) \tag{3.11}$$

Some caution must be taken when applying Equations 3.10-11. Firstly, these equations require generally that T and T^{Ref} be entered as absolute temperatures, in K rather than °C. Secondly, the thermal response parameters are often tabulated on a per-mole basis rather than a per-mass (specific) basis [59, 60].

The specific heat content of stream k is given by

$$w_{\mathrm{H}k}(T) = \frac{h_k(T)}{m_k}$$

Denoting $w_{\mathrm{H}k}^{\mathrm{Ref}} = w_{\mathrm{H}k}(T^{\mathrm{Ref}})$, it follows from Equation 3.9 that

$$w_{\mathrm{H}k}^{\mathrm{Ref}} = \sum_{j \in \mathcal{S}_k} w_{\mathrm{H}j}^{\mathrm{Ref}} w_{jk} \tag{3.12}$$

Applying the Fundamental Theorem of Calculus,

$$w_{\mathrm{H}k}(T) = w_{\mathrm{H}k}^{\mathrm{Ref}} + \int_{T^{\mathrm{Ref}}}^{T} c_k(T')dT'$$

where c_k is the specific heat capacity of k, i.e. the partial derivative of $w_{\mathrm{H}k}$ with respect to T.

From Equations 3.9-10, it follows that

$$c_k(T) = A_k + B_k T + C_k T^{-2} + D_k T^2 \tag{3.13}$$

where

$$A_k = \sum_{j \in \mathcal{S}_k} A_j w_{jk} \tag{3.14}$$

$$B_k = \sum_{j \in \mathcal{S}_k} B_j w_{jk} \tag{3.15}$$

$$C_k = \sum_{j \in \mathcal{S}_k} C_j w_{jk} \tag{3.16}$$

$$D_k = \sum_{j \in \mathcal{S}_k} D_j w_{jk} \tag{3.17}$$

The integration gives

$$w_{\mathrm{H}k}(T) = w_{\mathrm{H}k}^{\mathrm{Ref}} + A_k(T - T^{\mathrm{Ref}}) + \frac{B_k}{2}\left(T^2 - (T^{\mathrm{Ref}})^2\right) - C_k\left(T^{-1} - (T^{\mathrm{Ref}})^{-1}\right) + \frac{D_k}{3}\left(T^3 - (T^{\mathrm{Ref}})^3\right) \tag{3.18}$$

Collectively, $w_{\mathrm{H}k}^{\mathrm{Ref}}$ and (A_k, B_k, C_k, D_k) are the temperature response parameters of stream k. If the temperature T and the species composition $\{w_{jk} |\ j \in \mathcal{S}_k\}$ are available, then thermochemical data (e.g. Table 3.1) can be used in conjunction with Equations 3.12 and 3.14-18 to compute the specific heat of stream k.

A certain convention is maintained, regarding the specific heat contents. When $w_{\mathrm{H}j}$ or $w_{\mathrm{H}k}$ is written without a superscript, it implies a function of the temperature, be it $w_{\mathrm{H}j}(T)$ or $w_{\mathrm{H}k}(T)$. Otherwise the superscript describes a particular temperature, as in $w_{\mathrm{H}j}^{\mathrm{Ref}} = w_{\mathrm{H}j}(T^{\mathrm{Ref}})$ or $w_{\mathrm{H}k}^{\mathrm{Ref}} = w_{\mathrm{H}k}(T^{\mathrm{Ref}})$, evaluated at the reference temperature T^{Ref}. More generally, $w_{\mathrm{H}j}^{\mathrm{Superscript}} = w_{\mathrm{H}j}(T^{\mathrm{Superscript}})$.

A higher temperature corresponds to more heat, so it is intuitive that $w_{\mathrm{H}j}$ and $w_{\mathrm{H}k}$ should be strictly increasing functions of temperature. Thus c_j and c_k are inherently positive. (In actuality there are some exotic circumstances in which $w_{\mathrm{H}j}$ and $w_{\mathrm{H}k}$ actually decrease with temperature [64], but such cases are beyond the scope of pyrometallurgy). The positivity of c_j and c_k is especially important to the MILP formulation since it allows temperature bounds to be implemented indirectly, as heat bounds.

There may be some temperature bounds, $\underline{T}$ and $\overline{T}$, that must be satisfied by stream k such that $\underline{T} \leq T \leq \overline{T}$. These temperature bounds are converted into heat bounds using Equations 3.9 and 3.11. The temperature bounds are satisfied if, and only if, the corresponding heat bounds are satisfied.

3.2.4 Heat Distribution Across Several Process Streams

The bath of a Peirce-Smith converter can contain several streams that are in thermal contact with each other. This may include a combination of product streams, and perhaps some newly added feed streams that have not yet disintegrated. Thermal equilibrium is held between these streams as they approach a common temperature.

A newly added charge will undoubtedly create a spatial distribution of temperature within the bath. However, the bath becomes well-mixed as soon as a blowing action begins, except for the local disturbances surrounding the incoming blast [7], which are neglected in this study. In any case, these disturbances subside as soon as the blowing action is halted. Thus it is assumed that during and following any blowing action, there is a uniform temperature that extends throughout the bath, and that this temperature corresponds to the equilibrium temperature.

The equilibrium temperature T depends on the amount of heat h which is contained in the bath,

$$h = \sum_{k \in \mathcal{Z}_{\mathrm{NG}}} [w_{\mathrm{H}k}(T)] m_k \tag{3.19}$$

Again, this temperature is not necessarily uniform throughout the bath, except during and

after the blowing actions. The summation in Equation 3.19 considers only the nongaseous species, since the gaseous species are not retained within the bath.

In practice h can be estimated, and T can then be computed from Equation 3.19 by applying Newton's Method [13]. Given T, Equation 3.9 determines how h will be distributed among the constituent streams at the onset of a blowing action. It can be verified that

$$h = \sum_{k \in \mathcal{Z}_{\mathrm{NG}}} h_k(T)$$

so that the heat is indeed distributed into $\mathcal{Z}_{\mathrm{NG}}$.

To implement Newton's Method it is convenient to consider a global specific heat content,

$$w_{\mathrm{H}} = \frac{h}{m}$$

where $m = \sum_{k \in \mathcal{Z}_{\mathrm{NG}}} m_k$ is the total mass of the bath. By substituting Equation 3.18 into Equation 3.19,

$$w_{\mathrm{H}} = w_{\mathrm{H}}^{\mathrm{Ref}} + A(T - T^{\mathrm{Ref}}) + \frac{B}{2}\left(T^2 - (T^{\mathrm{Ref}})^2\right) - C\left(T^{-1} - (T^{\mathrm{Ref}})^{-1}\right) + \frac{D}{3}\left(T^3 - (T^{\mathrm{Ref}})^3\right) \quad (3.20)$$

in which $(w_{\mathrm{H}}^{\mathrm{Ref}}, A, B, C, D)$ are global temperature response parameters,

$$w_{\mathrm{H}}^{\mathrm{Ref}} = \left(\sum_{k \in \mathcal{Z}_{\mathrm{NG}}} w_{\mathrm{H}k}^{\mathrm{Ref}} w_k\right) \quad (3.21)$$

$$A = \left(\sum_{k \in \mathcal{Z}_{\mathrm{NG}}} A_k w_k\right) \quad (3.22)$$

$$B = \left(\sum_{k \in \mathcal{Z}_{\mathrm{NG}}} B_k w_k\right) \quad (3.23)$$

$$C = \left(\sum_{k \in \mathcal{Z}_{\mathrm{NG}}} C_k w_k\right) \quad (3.24)$$

$$D = \left(\sum_{k \in \mathcal{Z}_{\mathrm{NG}}} D_k w_k\right) \quad (3.25)$$

where $w_k = m_k/m$ is the mass fraction of stream k in the bath. Given w_{H} and $(w_{\mathrm{H}}^{\mathrm{Ref}}, A, B, C, D)$,

Newton's Method reduces to a fixed point iteration [65],

$$T \leftarrow \left(T + \frac{w_{\mathrm{H}} - w_{\mathrm{H}}^{\mathrm{Ref}} - A(T - T^{\mathrm{Ref}}) - \frac{B}{2}\left(T^2 - (T^{\mathrm{Ref}})^2\right) + C\left(T^{-1} - (T^{\mathrm{Ref}})^{-1}\right) - \frac{D}{3}\left(T^3 - (T^{\mathrm{Ref}})^3\right)}{A + BT + CT^{-2} + DT^2} \right) \tag{3.26}$$

A guessed value for T is substituted into the righthand side, in order to compute an improved guessed value; the improved value is then recycled into the righthand side to give an even better value, and so on, until two subsequent guesses are sufficiently close (e.g. equal to 8 significant digits).

The denominator in Equation 3.26 is effectively the global specific heat capacity. It is strictly nonnegative in the temperature range of interest, which follows from the nonnegativity of the constituent specific heat capacities c_k. This condition causes Equation 3.26 to converge to a single value, assuming that that the first guess is sufficiently close to the true value [13]. $T = 1475$ K works well as a first guess since it is fairly representative of actual PS bath temperatures.

Equations 3.20 and 3.26 describe the one-to-one correspondence between w_{H} and T, both of which are essential measures of heat concentration. The specific heat content w_{H} is the global ratio of heat per mass. On the other hand, the temperature T controls how much heat is allocated to each unit mass, depending on the chemical nature of these unit masses.

3.3 Characterization of Feed Streams

3.3.1 Furnace Matte

Mattes are simpler to characterize than the other feeds since they are presumably dominated by sulfides. The valuable metal grades ($w_{\mathrm{Ni,FMatte}}, w_{\mathrm{Co,FMatte}}, w_{\mathrm{Cu,FMatte}}$) are usually known, which is sufficient to complete the missing composition information, and to predict the volume and temperature response.

Assuming that the furnace matte is composed only of the aforementioned sulfides, it follows that $\mathcal{E}_{\mathrm{FMatte}} = \{\mathrm{Fe, Ni, Co, Cu, S}\}$ and $\mathcal{S}_{\mathrm{FMatte}} = \{\mathrm{FeS, Ni_3S_2, CoS, Cu_2S}\}$. Therefore,

$$\begin{aligned} w_{\mathrm{Fe,FMatte}} + w_{\mathrm{Ni,FMatte}} + w_{\mathrm{Co,FMatte}} + w_{\mathrm{Cu,FMatte}} + w_{\mathrm{S,FMatte}} &= 1 \\ w_{\mathrm{FeS,FMatte}} + w_{\mathrm{Ni_3S_2,FMatte}} + w_{\mathrm{CoS,FMatte}} + w_{\mathrm{Cu_2S,FMatte}} &= 1 \end{aligned}$$

and Equation 3.5 gives

$$\begin{aligned}
w_{\mathrm{Fe,FMatte}} &= w_{\mathrm{Fe,FeS}} w_{\mathrm{FeS,FMatte}} \\
w_{\mathrm{Ni,FMatte}} &= w_{\mathrm{Ni,Ni_3S_2}} w_{\mathrm{Ni_3S_2,FMatte}} \\
w_{\mathrm{Co,FMatte}} &= w_{\mathrm{Co,CoS}} w_{\mathrm{CoS,FMatte}} \\
w_{\mathrm{Cu,FMatte}} &= w_{\mathrm{Cu,Cu_2S}} w_{\mathrm{Cu_2S,FMatte}} \\
w_{\mathrm{S,FMatte}} &= w_{\mathrm{S,FeS}} w_{\mathrm{FeS,FMatte}} + w_{\mathrm{S,Ni_3S_2}} w_{\mathrm{Ni_3S_2,FMatte}} + w_{\mathrm{S,CoS}} w_{\mathrm{CoS,FMatte}} + w_{\mathrm{S,Cu_2S}} w_{\mathrm{Cu_2S,FMatte}}
\end{aligned}$$

This constitutes 7 equations, 6 of which are linearly independent. This is fitting since there are exactly 6 unknown mass-fractions.

There are two unknown elemental fractions, $w_{\mathrm{Fe,FMatte}}$ and $w_{\mathrm{S,FMatte}}$, determined by substitution,

$$w_{\mathrm{Fe,FMatte}} = w_{\mathrm{Fe,FeS}} - \left(\frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Ni,Ni_3S_2}}}\right) w_{\mathrm{Ni,FMatte}} - \left(\frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Co,CoS}}}\right) w_{\mathrm{Co,FMatte}} - \left(\frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Cu,Cu_2S}}}\right) w_{\mathrm{Cu,FMatte}} \tag{3.27}$$

$$\begin{aligned}
w_{\mathrm{S,FMatte}} &= 1 - w_{\mathrm{Fe,FeS}} - \left(1 + \frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Ni,Ni_3S_2}}}\right) w_{\mathrm{Ni,FMatte}} - \left(1 + \frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Co,CoS}}}\right) w_{\mathrm{Co,FMatte}} \\
&\quad - \left(1 + \frac{w_{\mathrm{Fe,FeS}}}{w_{\mathrm{Cu,Cu_2S}}}\right) w_{\mathrm{Cu,FMatte}}
\end{aligned} \tag{3.28}$$

thus completing the species composition $\{w_{ik} |\ i \in \mathcal{E}_{\mathrm{FMatte}}\}$. Again by substitution,

$$w_{\mathrm{FeS,FMatte}} = 1 - \frac{w_{\mathrm{Ni,FMatte}}}{w_{\mathrm{Ni,Ni_3S_2}}} - \frac{w_{\mathrm{Co,FMatte}}}{w_{\mathrm{Co,CoS}}} - \frac{w_{\mathrm{Cu,FMatte}}}{w_{\mathrm{Cu,Cu_2S}}} \tag{3.29}$$

$$w_{\mathrm{Ni_3S_2,FMatte}} = \frac{w_{\mathrm{Ni,FMatte}}}{w_{\mathrm{Ni,Ni_3S_2}}} \tag{3.30}$$

$$w_{\mathrm{CoS,FMatte}} = \frac{w_{\mathrm{Co,FMatte}}}{w_{\mathrm{Co,CoS}}} \tag{3.31}$$

$$w_{\mathrm{Cu_2S,FMatte}} = \frac{w_{\mathrm{Cu,FMatte}}}{w_{\mathrm{Cu,Cu_2S}}} \tag{3.32}$$

thus completing the species composition $\{w_{jk} |\ j \in \mathcal{S}_{\mathrm{FMatte}}\}$.

Equations 3.27-32 describe furnace matte for the General Nickel-Copper PSC Formulation. For the Simplified Copper PSC Formulation, it suffices to set $w_{\mathrm{Ni,FMatte}} = w_{\mathrm{Co,FMatte}} = 0$. Given the species composition, the density can be estimated using Equation 3.8, and the

temperature response parameters using Equations 3.12 and 3.14-17.

3.3.2 Fluxes and Secondary Feeds

In Table 3.2, the characterization of fluxes and secondary feeds are considered with three levels of detail. It is common in industry that the only available composition data for a feed k is its elemental mass-fractions $\{w_{ik} |\ i \in \mathcal{E}_k\}$; this description falls into Level 1 of Table 3.2, and is insufficient because there is no general way of anticipating the volume and temperature response.

For certain feeds, the species composition $\{w_{jk} |\ j \in \mathcal{S}_k\}$ is available. In particular, fluxes are usually a known blend of the stable oxides. Assuming that the fluxes are fed at ambient temperature (roughly 30°C for a smelter), this qualifies as Level 2.

Given the species composition, the results of Subsections 3.2.2 and 3.2.3 are used to estimate whichever response parameters may be missing in a Level 2 description. Of course, Equation 3.5 is a straight-forward means of converting $\{w_{jk} |\ j \in \mathcal{S}_k\}$ into $\{w_{ik} |\ i \in \mathcal{E}_k\}$. Thus Level 2 can be upgraded to Level 3, as depicted in Figure 3.2.

A Level 3 characterization includes all of the necessary data to assess heat and elemental mass contributions, as well as temperature and volume responses. The species composition $\{w_{jk} |\ j \in \mathcal{S}_k\}$ is not strictly necessary, as long as the density and all of the temperature response parameters are otherwise available. In preliminary plant studies, it is unlikely that Level 3 data is directly available for all of the feeds, especially not for the secondary feeds [55, 66]. However, more advanced studies can include the appropriate experimental trials and measurements that lead to Level 3 data.

In preliminary calculations, Level 1 is usually all that is available for secondary feeds.

Table 3.2: Description levels for fluxes and secondary feeds

1) **Element-Based Characterization** (Insufficient)	
Composition:	all of $\{w_{ik} |\ i \in \mathcal{E}_k\}$
Volume and Temperature Response:	**not** all of $\{\rho_k, w_{\mathrm{H}k}^{\mathrm{Ref}}, A_k, B_k, C_k, D_k\}$
2) **Species-Based Characterization** (Sufficient)	
Composition:	all of $\{w_{jk} |\ j \in \mathcal{S}_k\}$
Volume and Temperature Response:	**not** all of $\{\rho_k, w_{\mathrm{H}k}^{\mathrm{Ref}}, A_k, B_k, C_k, D_k\}$
3) **Necessary Characterization**	
Mass Composition:	all of $\{w_{ik} |\ i \in \mathcal{E}_k\}$
Volume and Temperature Response:	all of $\{\rho_k, w_{\mathrm{H}k}^{\mathrm{Ref}}, A_k, B_k, C_k, D_k\}$

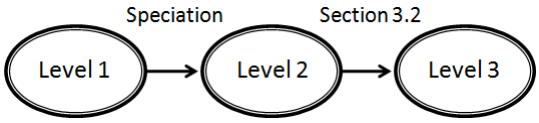


Figure 3.2: Level 1 is upgraded to Level 2 using a speciation technique, and Level 2 is upgraded to Level 3 using the results of Section 3.2

Table 3.3: Elemental composition of a revert stream k [66]

i	w_{ik}
Fe	0.2404
Cu	0.5015
S	0.0618
Si	0.0599
O	0.1365

Figure 3.2 shows the importance of speciation, which allows Level 1 to be upgraded to Level 2, which can in turn be upgraded to Level 3. Speciation procedures usually require so-called expert knowledge about the origins of the stream and the species which are most likely to be present. Thus expert knowledge gives an expectation of $\mathcal{S}_k$, which is denoted $\hat{\mathcal{S}}_k$.

For example, $k \in \mathcal{Z}_{\text{Feed}}$ might represent a revert stream from a copper smelter, whose elemental composition is given in Table 3.3, taken from [66]. Copper reverts are generally a mixture of flux, slag, matte and/or blister copper, which originate from bath spills, splashes, the cleaning of equipment, or the accumulation of flue dust. Supposing that this particular stream were produced through the accumulation of dust, it is likely to be oxidized, so that $\hat{\mathcal{S}}_k = \{Fe_2SiO_{4(sol)}, Fe_3O_{4(sol)}, Cu_{(sol)}, Cu_2O_{2(sol)}\}$. (It is reasonable to consider $Cu_{(sol)}$ since metallic copper is an oxidation product of Cu_2S, as per the Copper-Blow). Table 3.4 contains the density and temperature response parameters for $\hat{\mathcal{S}}_k$.

Given $\{w_{ik} |\ i \in \mathcal{E}_k\}$ and $\hat{\mathcal{S}}_k$, the species composition $\{w_{jk} |\ j \in \mathcal{S}_k\}$ is approximated by the values $\{\hat{w}_{jk} |\ j \in \hat{\mathcal{S}}_k\}$ which minimize the Euclidean distance [67],

$$D = \sqrt{\sum_{i \in \mathcal{E}_k} \left(w_{ik} - \sum_{j \in \hat{\mathcal{S}}_k} w_{ij} \hat{w}_{jk} \right)^2}$$

while satisfying $\sum_{j \in \hat{\mathcal{S}}_k} \hat{w}_{jk} = 1$, and $\hat{w}_{jk} \geq 0$ for all $j \in \hat{\mathcal{S}}_k$. This speciation technique is known as Convex Projection [54, 55]. Figure 3.3 illustrates the concept of Convex Projection for $|\mathcal{E}_k| = 3$, although the current MILP implies up to eleven dimensions, $|\mathcal{E}| = 11$.

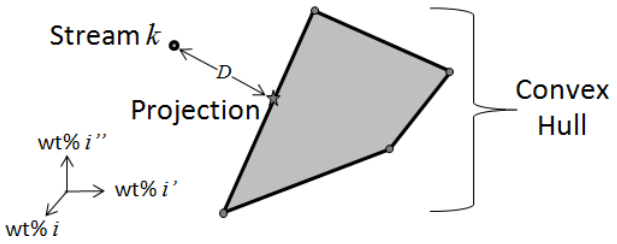


Figure 3.3: A stream k, composed of 3 elements $\mathcal{E}_k = \{i, i', i''\}$, is projected into the convex hull of 4 species

The minimization of D is equivalent to the minimization of D^2, so that $\{\hat{w}_{jk} |\ j \in \hat{\mathcal{S}}_k\}$ is the result of a quadratic program (QP) [68]. Thus,

$$\sum_{i\in\mathcal{E}_k}\left(w_{ik} - \sum_{j\in\hat{\mathcal{S}}_k} w_{ij}\hat{w}_{jk}\right)^2 \quad = \quad \min_{w'_{jk}}\left(\quad \sum_{i\in\mathcal{E}_k}\left(w_{ik} - \sum_{j\in\hat{\mathcal{S}}_k} w_{ij}w'_{jk}\right)^2\right) \tag{3.33}$$

$$\text{such that} \qquad \sum_{j\in\hat{\mathcal{S}}_k} w'_{jk} = 1,$$

$$w'_{jk} \geq 0 \text{ for all } j \in \hat{\mathcal{S}}_k$$

Indeed, $\{\hat{w}_{jk} |\ j \in \hat{\mathcal{S}}_k\}$ must itself satisfy $\sum_{j\in\hat{\mathcal{S}}_k} \hat{w}_{jk} = 1$, and $\hat{w}_{jk} \geq 0$ for all $j \in \hat{\mathcal{S}}_k$.

After applying the Convex Projection in conjunction with a QP solution technique (the Active Set Method [68], for example), the results of Sections 3.2 are used to compute the missing volume and temperature response parameters; $\{\hat{w}_{jk} |\ j \in \hat{\mathcal{S}}_k\}$ is used in lieu of $\{w_{jk} |\ j \in \mathcal{S}_k\}$, and the projected response parameters are denoted $(\hat{\rho}_k, \hat{w}^{\text{Ref}}_{\text{H}k}, \hat{A}_k, \hat{B}_k, \hat{C}_k, \hat{D}_k)$.

Table 3.4: Density and temperature response parameters [59, 60] for $j \in \hat{\mathcal{S}}_k$, where k is a revert stream that was formed by the accumulation of flue dust, having the elemental composition given in Table 3.3

j	ρ_j [g/L]	$w^{\text{Ref}}_{\text{H}j}$ [J/g]	A_j [J/(g°C)]	B_j [J/(g(°C)^2)]	C_j [(J°C)/g]	D_j [J/(g(°C)^3)]	$\hat{w}_{jk}$
$Cu_2S_{(sol)}$	5.8	-498.91	0.24669	$8.2055{\cdot}10^{-4}$	0	0	0.30682
$Cu_{(sol)}$	8.92	0	0.35834	$9.6303{\cdot}10^{-5}$	0	0	0.25641
$Fe_2SiO_{4(sol)}$	4.392	-7.2540	0.68958	$3.9173{\cdot}10^{-4}$	-18070.	0	0.43204
$Fe_3O_{3(sol)}$	5.2	-4825.2	0.74430	$3.4024{\cdot}10^{-4}$	-17708.	0	0.00473
$Cu_2O_{(sol)}$	6	-1257.9	0.43567	$1.6668{\cdot}10^{-4}$	0	0	0
$SiO_{2(sol)}$	2.65	-14166.	0.75726	$6.0693{\cdot}10^{-4}$	-16803.	0	0
Projection	5.52	-172.76	0.46902	$4.4730{\cdot}10^{-4}$	-7890.9	0	

In Table 3.4, the rightmost column shows the projected speciation data for the flue dust example, and the bottom line has the corresponding response parameters. These particular results were obtained using the Microsoft Excel© Solver [69].

It is preferable to minimize the use of projected data, in favour of actual measured data. For example, if a reliable measurement of ρ_k is available, then there is no need to consider $\hat{\rho}_k$. Also, the mass contribution should be based on the original composition data $\{w_{ik} |\ i \in \mathcal{E}_k\}$, rather than the projected elemental composition $\{\sum_{j \in \hat{\mathcal{S}}_k} w_{ij}\hat{w}_{jk} |\ i \in \mathcal{E}_k\}$.

Convex Projection allows the MILP formulation to be applied even in preliminary smelter studies, since it estimates the response parameters of poorly characterized secondary feeds. This approach is not usually necessary for fluxes that tend to be characterized using a species-based composition, nor is it necessary for mattes which are dominated by sulfides.

3.3.3 Blast

The blast is injected into the converter through a set of tuyeres at a rate that is approximately constant. In industry, the blast rate and blast composition are expressed on a volumetric basis. These two parameters are used to compute the oxygen mass rate (T/h of oxygen), hence to estimate the time required to convert a given feed tonnage. A third industrial parameter is the blast temperature, which affects the heat balance.

The blast rate is denoted $\dot{v}_{\text{Blast}}$, in which the dot implies a derivative with respect to a duration of time. The volumetric composition is given by $\{\phi_j |\ j \in \mathcal{S}_{\text{Blast}}\}$ in which ϕ_j is the volume fraction of species j contained in the blast.

Currently, PS blasts are composed of only two species, namely oxygen gas and nitrogen gas; thus $\mathcal{S}_{\text{Blast}} = \{O_2,\ N_2\}$, and it follows that $\mathcal{E}_{\text{Blast}} = \{O,\ N\}$. With this simple two-element configuration, the speciation is trivial; Equation 3.5 gives $w_{\text{O,Blast}} = w_{\text{O}_2\text{,Blast}}$ and $w_{\text{N,Blast}} = w_{\text{N}_2\text{,Blast}}$. Multiplying through by m_{Blast}, it follows that $m_{\text{O,Blast}} = m_{\text{O}_2\text{,Blast}}$ and $m_{\text{N,Blast}} = m_{\text{N}_2\text{,Blast}}$.

A more complicated blast configuration is conceivable. For instance, some researchers have considered using SO_2 to replace some of the nitrogen, hence favouring subsequent acid production [42]; sulfur would then be an additional blast element. Nonetheless, this section only considers the two-element configuration, which is currently used in all PS operations worldwide.

Considering that O_2 is the only oxygen bearing species under the two-element model, the

oxygen mass-rate is given by

$$\dot{m}_{\mathrm{O,Blast}} = \rho_{\mathrm{O}_2}\dot{v}_{\mathrm{Blast}}\phi_{\mathrm{O}_2}$$

where ϕ_{O_2} is the oxygen enrichment, although it is expressed as a fraction rather than a percentage.

As discussed in Subsection 1.3.1, the volumetric quantities ρ_{O_2} and $\dot{v}_{\mathrm{Blast}}$ are stated with respect to Normal Conditions, meaning atmospheric pressure and a temperature of 0°C. Thus the superscript "Norm" is introduced,

$$\dot{m}_{\mathrm{O,Blast}} = \rho_{\mathrm{O}_2}^{\mathrm{Norm}}\dot{v}_{\mathrm{Blast}}^{\mathrm{Norm}}\phi_{\mathrm{O}_2} \tag{3.34}$$

where $\rho_{\mathrm{O}_2}^{\mathrm{Norm}}$ and $\dot{v}_{\mathrm{Blast}}^{\mathrm{Norm}}$ are respectively the density of oxygen gas and the blast rate, both measured under Normal Conditions. The density $\rho_{\mathrm{O}_2}^{\mathrm{Norm}}$ is given by Equation 3.7,

$$\rho_{\mathrm{O}_2}^{\mathrm{Norm}} = 0.0014276 \text{ (T/Nm}^3\text{)}$$

To use Equation 3.34, $\dot{v}_{\mathrm{Blast}}^{\mathrm{Norm}}$ should be given in Normal units, such as Nm^3/h or Nm^3/min.

The two-element model assumes that

$$\phi_{\mathrm{O}_2} + \phi_{\mathrm{N}_2} = 1$$

so that the blast volume is entirely composed of oxygen and nitrogen, and N_2 is the only nitrogen bearing species. Thus the nitrogen mass-rate is given by

$$\dot{m}_{\mathrm{N,Blast}} = \rho_{\mathrm{N}_2}\dot{v}_{\mathrm{Blast}}\phi_{\mathrm{N}_2} = \rho_{\mathrm{N}_2}\dot{v}_{\mathrm{Blast}}(1-\phi_{\mathrm{O}_2})$$

Measuring under Normal Conditions,

$$\dot{m}_{\mathrm{N,Blast}} = \rho_{\mathrm{N}_2}^{\mathrm{Norm}}\dot{v}_{\mathrm{Blast}}^{\mathrm{Norm}}(1-\phi_{\mathrm{O}_2}) \tag{3.35}$$

where $\dot{v}_{\mathrm{Blast}}^{\mathrm{Norm}}$ should be measured in Normal units. Using Equation 3.7,

$$\rho_{\mathrm{N}_2}^{\mathrm{Norm}} = 0.0012498 \text{ (T/Nm}^3\text{)}$$

Under the two-element blast model, Equations 3.34-35 provide the mass-rate at which O and N are blown into the converter as part of the blast. For all remaining elements, $\dot{m}_{i\mathrm{Blast}} = 0$ when $i \notin \{\mathrm{O,N}\}$.

The rate at which heat is blown into the system, as part of the blast, is related to the blast temperature T^{Blast},

$$\dot{h}_{\text{Blast}} = \sum_{j \in \mathcal{S}_{\text{Blast}}} w_{\text{H}j}^{\text{Blast}} \dot{m}_{j\text{Blast}}$$

where the specific heat contents $w_{\text{H}j}^{\text{Blast}} = w_{\text{H}j}(T^{\text{Blast}})$ are calculated using Equation 3.11 and Table 3.1. Factoring out $\dot{v}_{\text{Blast}}^{\text{Norm}}$, it follows that

$$\dot{h}_{\text{Blast}} = \left(\sum_{j \in \mathcal{S}_{\text{Blast}}} w_{\text{H}j}^{\text{Blast}} \rho_j^{\text{Norm}} \phi_j \right) \dot{v}_{\text{Blast}}^{\text{Norm}}$$

where ϕ_j is the volume fraction of species j in the blast. For the two-element model,

$$\dot{h}_{\text{Blast}} = \left(w_{\text{H,O}_2}^{\text{Blast}} \rho_{\text{O}_2}^{\text{Norm}} \phi_{\text{O}_2} + w_{\text{H,N}_2}^{\text{Blast}} \rho_{\text{N}_2}^{\text{Norm}} [1 - \phi_{\text{O}_2}] \right) \dot{v}_{\text{Blast}}^{\text{Norm}} \tag{3.36}$$

using the same densities, $\rho_{\text{O}_2}^{\text{Norm}}$ and $\rho_{\text{N}_2}^{\text{Norm}}$, as for Equations 3.34-35.

Equation 3.34 is essential in controlling the oxygenation of the charge, and Equation 3.36 is essential in controlling the temperature of the charge. These equations make direct use of three industrial blast parameters, namely the blast rate $\dot{v}_{\text{Blast}}^{\text{Norm}}$, the oxygen enrichment ϕ_{O_2}, and the blast temperature T^{Blast}.

3.4 Characterization of Product Streams

3.4.1 Regime-Dependence of Product Species

A PS reaction regime is identified by the reaction that occurs under marginal additions of oxygen. For example, if a small addition of oxygen would contribute to the Slag-Blow reaction, then the system is in the Slag-Blow regime. Otherwise, if the small addition of oxygen would contribute to the Cobalt-Overblow, then the system is in the Cobalt-Overblow regime, etc.

In the General Nickel-Copper PS, the system may pass from the Slag-Blow regime into the Nickel-Overblow, into the Cobalt-Overblow, into the Copper-Blow and into the Copper-Overblow regime; the ordering follows from the relative stabilities of the sulfide species FeS, Ni_3S_2, CoS and Cu_2S, and the stability of $Cu_{(liq)}$ that persists throughout the Copper-Blow and the Copper-Overblow. The simplified copper PS systems proceed directly from the Slag-Blow regime into the Copper-Blow, since there are no nickel and cobalt sulfides to participate in the reactions.

Following this discussion, the set of reaction regimes is written,

$$\mathcal{R} \;=\; \{\;\; \text{SlagBlow, NickelOverblow, CobaltOverblow, CopperBlow, CopperOverblow} \;\;\}$$

for the General Nickel-Copper PSC Formulation. However, the NickelOverblow and CobaltOverblow are omitted from the Simplified Copper PSC Formulation.

The set $\mathcal{S}_{\text{NGProd}}$ represents all of the species that can report to the nongaseous product streams. However, the occurrence of certain species depends on the reaction regimes. Tables 3.5 and 3.6 describes the various species as a function of the regime in the General Nickel-Copper PSC Formulation and in the Simplified Copper PSC Formulation, respectively. Every species in $\mathcal{S}_{\text{NGProd}}$ occurs in at least one reaction regime.

The tables are constructed using the following preliminary considerations.

- FeS is not stable in the presence of NiO, CoO, $Cu_{(liq)}$ or Cu_2O since sulfur would migrate away from FeS in favour of more stable sulfides.
- Ni_3S_2 is not stable in the presence of CoO, $Cu_{(liq)}$ or Cu_2O since sulfur would migrate away from Ni_3S_2 in favour of more stable sulfides.
- CoS is not stable in the presence of $Cu_{(liq)}$ or Cu_2O since sulfur would migrate away from CoS in favour of Cu_2S.
- Cu_2S is not stable in the presence of Cu_2O since sulfur would migrate away from Cu_2S, as oxygen migrates away from Cu_2O, resulting in $Cu_{(liq)}$ and SO_2.

Tables 3.5 and 3.6 consider only the regime-dependent product species. It can be verified that all of the other members of $\mathcal{S}_{\text{NGProd}}$ can occur under PS conditions, regardless of the reaction regimes. For example, the ferroslag species, Fe_2SiO_4 and Fe_3O_4, are produced during the Slag-Blow, and are not consumed in any of the other regimes, hence they can occur in all of the regimes. Similar observations can be made for the stable oxide species.

Table 3.5: Stability of regime-dependent species the General Nickel-Copper PSC Formulation

	FeS	Ni_3S_2	CoS	Cu_2S	$Cu_{(liq)}$	NiO	CoO	Cu_2O
Slag-Blow	√	√	√	√	×	×	×	×
Nickel-Overblow	×	√	√	√	×	√	×	×
Cobalt-Overblow	×	×	√	√	×	√	√	×
Copper-Blow	×	×	×	√	√	√	√	×
Copper-Overblow	×	×	×	×	√	√	√	√

Table 3.6: Stability of regime-dependent species in the Simplified Copper PSC Formulation

	FeS	Cu_2S	$Cu_{(liq)}$	Cu_2O
Slag-Blow	$\surd$	$\surd$	$\times$	$\times$
Copper-Blow	$\times$	$\surd$	$\surd$	$\times$
Copper-Overblow	$\times$	$\times$	$\surd$	$\surd$

As discussed in Subsection 1.3.2, the Slag-Blow reaction requires FeS as a feed species; this precludes the occurrence of NiO, CoO, $Cu_{(liq)}$, and Cu_2O, thus giving the first line of Table 3.5 and of Table 3.6. The Nickel-Overblow reaction requires Ni_3S_2 and produces NiO, hence precluding FeS, CoO, $Cu_{(liq)}$, and Cu_2O. Similarly, the remaining rows of Tables 3.5 and 3.6 can be generated by considering the feeds and products of each of the reactions.

In every regime of the General Nickel-Copper PSC Formulation, there are four species in $\mathcal{S}_{\text{NGProd}}$ that are unstable, as shown in Table 3.5. Similarly, in every regime of the Simplified Copper PSC Formulation, there are two species in $\mathcal{S}_{\text{NGProd}}$ that are unstable, as shown in Table 3.6.

The set of regime-dependent product species is denoted $\mathcal{S}_{\text{RgProd}} \subset \mathcal{S}_{\text{NGProd}}$. Thus

$$\mathcal{S}_{\text{RgProd}} \;=\; \{\;\; \text{FeS, } Ni_3S_2\text{, CoS, } Cu_2S\text{, } Cu_{(liq)}\text{, NiO, CoO, } Cu_2O \;\;\}$$

in the general formulation. Naturally, the nickel and cobalt species are removed from the Simplified Copper PSC Formulation. For every regime-dependent species $j \in \mathcal{S}_{\text{RgProd}}$, there is a strict subset of regimes $\mathcal{R}_j \subsetneq \mathcal{R}$ for which this species is stable. Following the results of Table 3.5 for the General Nickel-Copper PSC Formulation,

$$\begin{aligned}
\mathcal{R}_{\text{FeS}} &= \{\; \text{SlagBlow} \;\} \\
\mathcal{R}_{Ni_3S_2} &= \{\; \text{SlagBlow, NickelOverblow} \;\} \\
\mathcal{R}_{\text{CoS}} &= \{\; \text{SlagBlow, NickelOverblow, CobaltOverblow} \;\} \\
\mathcal{R}_{Cu_2S} &= \{\; \text{SlagBlow, NickelOverblow, CobaltOverblow, CopperBlow} \;\} \\
\mathcal{R}_{Cu_{(liq)}} &= \{\; \text{CopperBlow, CopperOverblow} \;\} \\
\mathcal{R}_{\text{NiO}} &= \{\; \text{NickelOverblow, CobaltOverblow, CopperBlow, CopperOverblow} \;\} \\
\mathcal{R}_{\text{CoO}} &= \{\; \text{CobaltOverblow, CopperBlow, CopperOverblow} \;\} \\
\mathcal{R}_{Cu_2O} &= \{\; \text{CopperOverblow} \;\}
\end{aligned}$$

Similarly $\mathcal{R}_j$ can be constructed for simplified formulation, using Table 3.6.

By identifying the reaction regime of a PS system, several species can be eliminated

from the element-species mass balances of the product streams (Equation 3.3). This is an important step toward speciating the product streams, and ultimately determining whether or not the content of a converter would violate volumetric and thermal constraints.

3.4.2 Mass Distribution Across the Product Streams

The product streams are formed as the constituent species segregate from each other (Figure 3.1). This species-based segregation controls how the input mass is ultimately distributed into the product streams.

The product speciation balance is given by

$$m_{i\text{Prod}} = \sum_{j \in \mathcal{S}_{\text{Prod}}} w_{ij} m_{j\text{Prod}} \tag{3.37}$$

for all $i \in \mathcal{E}$, where $m_{i\text{Prod}}$ is the mass of element i that reports to the product streams, and $m_{j\text{Prod}}$ is the mass of species j that reports to the product streams. The element masses $m_{i\text{Prod}}$ are obtained by tallying up the contributions from the reacted feed streams,

$$m_{i\text{Prod}} = \sum_{k \in \mathcal{Z}_{\text{Feed}}} \xi_k w_{ik} m_k \tag{3.38}$$

for all $i \in \mathcal{E}$, where ξ_k is the portion of stream k that is reacted. In the context of the PSC Problem, ξ_k is taken as the weight-fraction of stream k that had been introduced into a converter prior to its previous blow, and has hence been reacted; this simplification is supported by the favourable mixing characteristics of PS converting [7].

If the species masses $m_{j\text{Prod}}$ could somehow be obtained from Equation 3.37, then the product tonnages and compositions would follow,

$$m_k = \sum_{j \in \mathcal{S}_k} m_{j\text{Prod}} \tag{3.39}$$

for all $k \in \mathcal{Z}_{\text{Prod}}$, and

$$w_{jk} = \frac{m_{j\text{Prod}}}{m_k} \tag{3.40}$$

for all $j \in \mathcal{S}_k$. The species composition given by Equation 3.40 can then be used to estimate the density and thermal response parameters using the results of Section 3.2, as was done for the feed streams.

However, Equation 3.37 is underspecified. For the General Nickel-Copper PSC Formula-

tion, there are $|\mathcal{S}_{\text{Prod}}| = 17$ unknown species masses $m_{j\text{Prod}}$, and $|\mathcal{E}| = 11$ independent linear equations, a difference of 6; considering that there are always 4 species that are excluded from the regime,

	17	unknown species
−	11	elemental balances
−	4	species excluded by regime
	2	degrees of freedom

thus leaving two degrees of freedom. For the Simplified Copper PSC Formulation, a similar analysis considers $|\mathcal{S}_{\text{Prod}}| = 13$, $|\mathcal{E}| = 9$ and the elimination of 2 species in every reaction regime,

	13	unknown species
−	9	elemental balances
−	2	species excluded by regime
	2	degrees of freedom

In either case, there are two degrees of freedom.

These two degrees of freedom, left over from Equation 3.37 and the reaction regimes, allow the consideration of two performance indicators. Firstly, the oxygen efficiency e_{O} is a common indicator within industry [7, 52, 66],

$$e_{\text{O}} = 1 - \left(\frac{m_{\text{O}_2,\text{Offgas}}}{m_{\text{O}_2,\text{Blast}}} \right)$$

It is the proportion of the blast oxygen $m_{\text{O}_2,\text{Blast}}$ that actually reacts with the bath, rather than exiting the system as unreacted O_2 within the offgas. Considering that all of the unreacted O_2 goes into the offgas, and that $m_{\text{O}_2,\text{Blast}} = m_{\text{O,Blast}}$, the equation can be rewritten as

$$e_{\text{O}} = 1 - \left(\frac{m_{\text{O}_2,\text{Prod}}}{m_{\text{O,Blast}}} \right) \tag{3.41}$$

which is more directly related to Equations 3.34 and 3.37. The oxygen efficiency is usually described as a percentage, which theoretically can range from 0% to 100%. In practice, e_{O} is between 90% and 100%, depending on the blast settings ($\dot{v}^{\text{Norm}}_{\text{Blast}}$ and ϕ_{O_2}), and several bath characteristics such as volume, temperature and viscosity [7]. The oxygen efficiency may also depend on the reaction regime, as the different reactions consume different proportions of oxygen.

The second performance indicator is the ferroslag ratio r_{FS}, which is the "FeO / Fe_3O_4" mass ratio. It quantifies the amount of blast oxygen required to hold iron into the slag. In the current formulation FeO is regarded as a component of the fayalite $2FeO{\cdot}SiO_2$, such that

$$r_{\mathrm{FS}} = \frac{m_{\mathrm{FeO,Prod}}}{m_{\mathrm{Fe_3O_4,Prod}}}$$

can be restated as

$$r_{\mathrm{FS}} = \left(\frac{2M_{\mathrm{FeO}}}{M_{\mathrm{Fe_2SiO_4}}}\right)\left(\frac{m_{\mathrm{Fe_2SiO_4,Prod}}}{m_{\mathrm{Fe_3O_4,Prod}}}\right) \tag{3.42}$$

As discussed in Subsection 1.3.2, fayalite Fe_2SiO_4 is preferable to magnetite Fe_3O_4 because it is a more efficient distribution of blast oxygen. Also, fayalite floats to the surface faster than magnetite, since it is a liquid. Theoretically, the ferroslag efficiency can range from 0 to infinity, but typical values range between 1 and 3.5 [7, 52]; the exact value depends on the amount of stable oxides {CaO, Al_2O_3, MgO} that accompany the silica flux [32].

The species composition of the product streams can be computed using Equations 3.37-40 if the reaction regime, the oxygen efficiency and the ferroslag ratio are known. The densities and the thermal parameters can then be obtained by applying the results of Section 3.2.

3.5 Flow Mechanisms

3.5.1 Streams, Actions and Flow Mechanisms

Table 3.7 classifies the feed and product streams with respect to the three converting actions that were introduced in Subsection 2.2.2, and distinguishes between the gaseous and nongaseous streams. This same decomposition is also used to characterize the mechanisms by which these streams are delivered to and from the converters.

Table 3.7: Decomposition of feed and product streams with respect to converting actions

		Actions			
		Charge	Blow	Discharge	Gaseous
Feeds	Charge	√	×	×	×
	Nongaseous Blow Feed	×	√	×	×
	Blast	×	√	×	√
Products	Offgas	×	√	×	√
	Discharge	×	×	√	×

Section 3.1 presented a division between feed and product streams, $\mathcal{Z} = \mathcal{Z}_{\mathrm{Feed}} \bigcup \mathcal{Z}_{\mathrm{Prod}}$. There is a second decomposition, $\mathcal{Z} = \mathcal{Z}_{\mathrm{Ch}} \bigcup \mathcal{Z}_{\mathrm{Blow}} \bigcup \mathcal{Z}_{\mathrm{DCh}}$, in which $\mathcal{Z}_{\mathrm{Ch}}$, $\mathcal{Z}_{\mathrm{Blow}}$ and $\mathcal{Z}_{\mathrm{DCh}}$ are

the streams which function during the charge, the blow and the discharge actions, respectively.

The charge streams are nongaseous feeds, $\mathcal{Z}_{\text{Ch}} \subset \mathcal{Z}_{\text{NGFeed}}$, while the discharge streams are nongaseous products, $\mathcal{Z}_{\text{DCh}} \subset \mathcal{Z}_{\text{NGProd}}$. (For simplicity, gas entrainment into the charge and discharge streams is ignored). However, the blow streams include nongaseous feeds, as well as the gaseous Blast and Offgas, so that $\mathcal{Z}_{\text{Blow}} = \mathcal{Z}_{\text{NGBlow}} \bigcup \mathcal{Z}_{\text{Blast}} \bigcup \{\text{Offgas}\}$, in which $\mathcal{Z}_{\text{NGBlow}}$ are the nongaseous streams that function during the blowing action.

The mechanics of Peirce-Smith converting prevents the removal of nongaseous products during the blow. Nongaseous feeds, however, can be introduced through Garr guns and chutes [7], hence $\mathcal{Z}_{\text{NGBlow}} \subset \mathcal{Z}_{\text{NGFeed}}$, as indicated in Table 3.7. $\mathcal{Z}_{\text{NGBlow}}$ typically includes flux and other granulated feeds.

The decomposition of Table 3.7 can be stated algebraically,

$$\begin{aligned} \mathcal{Z} &= \mathcal{Z}_{\text{Feed}} \bigcup \mathcal{Z}_{\text{Prod}} \\ \mathcal{Z}_{\text{Feed}} &= \mathcal{Z}_{\text{Ch}} \bigcup \mathcal{Z}_{\text{NGBlow}} \bigcup \mathcal{Z}_{\text{Blast}} \\ \mathcal{Z}_{\text{Prod}} &= \{\text{Offgas}\} \bigcup \mathcal{Z}_{\text{DCh}} \end{aligned}$$

Following the treatment of Section 3.1,

$$\begin{aligned} \mathcal{Z}_{\text{NGFeed}} &= \mathcal{Z}_{\text{Feed}} \setminus \mathcal{Z}_{\text{Blast}} &&= \mathcal{Z}_{\text{Ch}} \bigcup \mathcal{Z}_{\text{NGBlow}} \\ \mathcal{Z}_{\text{NGProd}} &= \mathcal{Z}_{\text{Prod}} \setminus \{\text{Offgas}\} &&= \mathcal{Z}_{\text{DCh}} \end{aligned}$$

Thus $\mathcal{Z}_{\text{NGProd}}$ and $\mathcal{Z}_{\text{DCh}}$ are identical. Indeed, nongaseous products are removed only during a discharge action.

Regarding $\mathcal{Z}_{\text{NGFeed}}$, it is plausible that there would be overlap between $\mathcal{Z}_{\text{Ch}}$ and $\mathcal{Z}_{\text{NGBlow}}$. For example, Flux $\in \mathcal{Z}_{\text{Ch}} \bigcap \mathcal{Z}_{\text{NGBlow}}$ if the flux may be fed prior to blow, as well as during the blow. However, $\mathcal{Z}_{\text{Ch}} \bigcap \mathcal{Z}_{\text{DCh}} = \varnothing$ and $\mathcal{Z}_{\text{NGBlow}} \bigcap \mathcal{Z}_{\text{DCh}} = \varnothing$ which is related to the fact that $\mathcal{Z}_{\text{Feed}} \bigcap \mathcal{Z}_{\text{Prod}} = \varnothing$.

$\mathcal{F}$ is the set of flow mechanisms, and each $j \in \mathcal{F}$ describes a technique for delivering a stream to or from a converter. These mechanisms undergo the same decomposition that is

depicted in Table 3.7 for streams,

$$\begin{aligned}\mathcal{F} &= \mathcal{F}_{\text{Feed}} \bigcup \mathcal{F}_{\text{Prod}} \\ \mathcal{F}_{\text{Feed}} &= \mathcal{F}_{\text{Ch}} \bigcup \mathcal{F}_{\text{NGBlow}} \bigcup \mathcal{F}_{\text{Blast}} \\ \mathcal{F}_{\text{Prod}} &= \mathcal{F}_{\text{Offgas}} \bigcup \mathcal{F}_{\text{DCh}}\end{aligned}$$

Thus $j \in \mathcal{F}_{\text{Feed}}$ represents a stream flowing into a converter. Similarly, $j \in \mathcal{F}_{\text{Prod}}$ is a stream flowing out of converter.

The flow mechanisms are related to the streams through the source mapping,

$$\text{srce} : \mathcal{F} \to \mathcal{Z}$$

such that $j \in \mathcal{F}$ is used to carry srce(j)$\in \mathcal{Z}$ to or from a converter. It follows that

$$\begin{aligned}\text{srce} &: \mathcal{F}_{\text{Ch}} \to \mathcal{Z}_{\text{Ch}} \\ &: \mathcal{F}_{\text{NGBlow}} \to \mathcal{Z}_{\text{NGBlow}} \\ &: \mathcal{F}_{\text{Blast}} \to \mathcal{Z}_{\text{Blast}} \\ &: \mathcal{F}_{\text{Offgas}} \to \{\text{Offgas}\} \\ &: \mathcal{F}_{\text{DCh}} \to \mathcal{Z}_{\text{DCh}}\end{aligned}$$

Thus $\mathcal{F}_{\text{Ch}}$, $\mathcal{F}_{\text{NGBlow}}$ and $\mathcal{F}_{\text{Blast}}$ carry the charge streams, nongaseous blow streams and blast into the converters, respectively. Similarly, $\mathcal{F}_{\text{Offgas}}$ and $\mathcal{F}_{\text{DCh}}$ carry the offgas and the discharge streams away from the converters.

As with streams, the nongaseous flows are treated separately $\mathcal{F}_{\text{NG}} = \mathcal{F}_{\text{NGFeed}} \bigcup \mathcal{F}_{\text{NGProd}}$, in which

$$\begin{aligned}\mathcal{F}_{\text{NGFeed}} &= \mathcal{F}_{\text{Feed}} \setminus \mathcal{F}_{\text{Blast}} &= \mathcal{F}_{\text{Ch}} \bigcup \mathcal{F}_{\text{NGBlow}} \\ \mathcal{F}_{\text{NGProd}} &= \mathcal{F}_{\text{Prod}} \setminus \mathcal{F}_{\text{Offgas}} &= \mathcal{F}_{\text{DCh}}\end{aligned}$$

Thus $\mathcal{F}_{\text{NGProd}}$ and $\mathcal{F}_{\text{DCh}}$ are identical, in the same way that $\mathcal{Z}_{\text{NGProd}}$ and $\mathcal{Z}_{\text{DCh}}$ are identical.

Formally, the srce mapping prevents any intersection between $\mathcal{F}_{\text{Ch}}$ and $\mathcal{F}_{\text{DCh}}$, or between $\mathcal{F}_{\text{NGBlow}}$ and $\mathcal{F}_{\text{DCh}}$, since $\mathcal{Z}_{\text{Ch}} \bigcap \mathcal{Z}_{\text{DCh}} = \mathcal{Z}_{\text{NGBlow}} \bigcap \mathcal{Z}_{\text{DCh}} = \varnothing$. Additionally, it is assumed that $\mathcal{F}_{\text{Ch}} \bigcap \mathcal{F}_{\text{NGBlow}} = \varnothing$ without any loss of generality; for $j \in \mathcal{F}_{\text{Ch}}$ and $j' \in \mathcal{F}_{\text{NGBlow}}$, the mere fact that j is active during charging, and j' during blowing, implies mechanistic differences between j and j' to the extent that $j \neq j'$. Thus an intersection between $\mathcal{F}_{\text{Ch}}$ and $\mathcal{F}_{\text{NGBlow}}$ is not permitted, even though there may well be an intersection between $\mathcal{Z}_{\text{Ch}}$ and $\mathcal{Z}_{\text{NGBlow}}$.

For any stream $k \in \mathcal{Z}$, there may be several flows $\mathcal{F}_k = \{j \in \mathcal{F} |\ \text{srce}(j) = k\}$ describing the modes by which k enters or leaves the converters. For example, there may be two sizes of feed ladles which both carry the same furnace matte, hence two different flows that draw on the same source.

The stream masses m_k have already been decomposed in terms of elements (Equation 3.1) and in terms of species (Equation 3.2). There is a now a third decomposition, with respect to the flow mechanisms,

$$m_k = \sum_{j \in \mathcal{F}_k} m^j \tag{3.43}$$

where m^j is the mass that is delivered by flow j. The flow mechanisms are specified in the superscript to avoid confusion with the species masses m_j.

A similar decomposition can be applied to the stream volume,

$$v_k = \sum_{j \in \mathcal{F}_k} v^j \tag{3.44}$$

where $v^j = m^j / \rho_{\text{srce}(j)}$ is the volume that is delivered by mechanism j. In the MILP formulation, it is convenient to consider the flow volumes v^j instead of the flow masses m^j. This is because ladles and other carrying devices impose volumetric constraints, as described in the following subsection.

Flow mechanisms are distinguished by which converting action is concurrent to their delivery, and by which stream they deliver. Additionally, they are distinguished by their dependence on ancillary objects, and their converter state requirements, which are all considered in the MILP formulation.

3.5.2 Modulated Charging and Discharging

Flows can be classified as modulated, semi-modulated or unmodulated. This classification determines how the various flows are implemented in the MILP.

A modulated flow mechanism implies discrete delivery units, such as pre-bundled packages of scrap, each having a fixed weight. A semi-modulated mechanism uses vessels that have a finite capacity, but are otherwise fed/drawn in arbitrary quantities; for example, matte is delivered using a discrete number of ladles, but the ladles may be partially filled, so that an arbitrary volume of matte can be delivered. Lastly, an unmodulated mechanism can be delivered in arbitrary quantities, free of the constraints imposed by discrete carrying vessels;

for example, the quantity of flux injected from Garr guns can be varied continuously.

The following disjoint decomposition applies for all charging flows,

$$\mathcal{F}_{\mathrm{Ch}} = \mathcal{F}_{\mathrm{Ch}}^{\mathrm{M}} \bigcup \mathcal{F}_{\mathrm{Ch}}^{\mathrm{SM}} \bigcup \mathcal{F}_{\mathrm{Ch}}^{\mathrm{UM}}$$

where $\mathcal{F}_{\mathrm{Ch}}^{\mathrm{M}}$, $\mathcal{F}_{\mathrm{Ch}}^{\mathrm{SM}}$ and $\mathcal{F}_{\mathrm{Ch}}^{\mathrm{UM}}$ represent the modulated, semi-modulated and unmodulated charge mechanisms. Similarly for discharging,

$$\mathcal{F}_{\mathrm{DCh}} = \mathcal{F}_{\mathrm{DCh}}^{\mathrm{M}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{SM}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{UM}}$$

where $\mathcal{F}_{\mathrm{DCh}}^{\mathrm{M}}$, $\mathcal{F}_{\mathrm{DCh}}^{\mathrm{SM}}$ and $\mathcal{F}_{\mathrm{DCh}}^{\mathrm{UM}}$ represent the modulated, semi-modulated and unmodulated discharge mechanisms. The remaining flows, $\mathcal{F}_{\mathrm{Blow}}$, are inherently unmodulated.

It is worthwhile to consider the unions of modulated and semi-modulated flow mechanisms,

$$\begin{aligned}
\mathcal{F}^{\mathrm{M}} &= \mathcal{F}_{\mathrm{Ch}}^{\mathrm{M}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{M}} \\
\mathcal{F}^{\mathrm{SM}} &= \mathcal{F}_{\mathrm{Ch}}^{\mathrm{SM}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{SM}} \\
\mathcal{F}_{\mathrm{Ch}}^{\mathrm{MSM}} &= \mathcal{F}_{\mathrm{Ch}}^{\mathrm{M}} \bigcup \mathcal{F}_{\mathrm{Ch}}^{\mathrm{SM}} \\
\mathcal{F}_{\mathrm{DCh}}^{\mathrm{MSM}} &= \mathcal{F}_{\mathrm{DCh}}^{\mathrm{M}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{SM}} \\
\mathcal{F}^{\mathrm{MSM}} &= \mathcal{F}_{\mathrm{Ch}}^{\mathrm{MSM}} \bigcup \mathcal{F}_{\mathrm{DCh}}^{\mathrm{MSM}}
\end{aligned}$$

because these are the mechanisms which involve discrete delivery units. For all $j \in \mathcal{F}^{\mathrm{MSM}}$, u^j is the number of units used to carry $\mathrm{srce}(j)$ via mechanism j.

In general, the delivery units place volumetric restrictions on the transport of charge and discharge streams. For modulated flows, the delivered volume is proportional to the quantity of delivery units,

$$v^j = v_u^j u^j \tag{3.45}$$

where v_u^j is the volume of $\mathrm{srce}(j)$ that is carried by a single unit of j, for all $j \in \mathcal{F}^{\mathrm{M}}$. The semi-modulated volumes are bounded by the number of delivery units,

$$v^j \leq \overline{v}_u^j u^j \tag{3.46}$$

where $\overline{v}_u^j$ is the maximum volume of $\mathrm{srce}(j)$ that can be carried by a single unit of j, for all $j \in \mathcal{F}^{\mathrm{SM}}$. Equations 3.45-46 are linear, as v_u^j and $\overline{v}_u^j$ are constant.

The previous equations can be rewritten in terms of delivery masses,

$$m^j = m_u^j u^j \tag{3.47}$$

where $m_u^j = \rho_{\mathrm{srce}(j)} v_u^j$, for all $j \in \mathcal{F}^{\mathrm{M}}$, and

$$m^j \leq \overline{m}_u^j u^j \tag{3.48}$$

where $\overline{m}_u^j = \rho_{\mathrm{srce}(j)} \overline{v}_u^j$, for all $j \in \mathcal{F}^{\mathrm{SM}}$.

Following the treatment of Section 3.3, the charge streams are assumed to have constant compositions, hence constant densities; in this case, Equations 3.46-47 are somewhat appealing for an MILP formulation, because they are linear, as m_u^j and $\overline{m}_u^j$ are constant. On the other hand, the discharge streams have a composition which depends on the chemical balance within the converter, (see Equations 3.37-40); thus Equations 3.47-48 are generally nonlinear for the discharge streams. The MILP formulation of Chapter 4 uses equations similar to 3.45-46, rather than 3.47-48, because the former work equally well for the charging mechanisms as for the discharging mechanisms.

The following MILP combines the dynamical description of Chapter 2 with the stream characterization of Chapter 3, within an optimization framework. Such a merger has not been accomplished before.

CHAPTER 4

MILP FORMULATION OF THE PSC PROBLEM

4.1 Gantt Structure

4.1.1 Assignments

The PSC Problem is posed as a mixed integer linear program (MILP) within this chapter. Numerical parameters and variables are defined, and are related to each other through linear constraints, and one of several objective functions that is to be optimized. The parameters, variables and constraints are described throughout the chapter, and the objective functions are described in Section 4.6.

The PSC Problem is to coordinate the various objects of a converting aisle, over a schedule which begins at time t_{Begin} and ends at time t_{End}. There may be some assignments which begin in the current schedule, and extend into the next schedule. The parameter $\bar{t}$ denotes the latest permissible completion time for any assignment planned for the current schedule, $t_{\text{Begin}} \leq t_{\text{End}} \leq \bar{t}$. To prevent the current assignments from spreading into the following schedule, $\bar{t}$ may be set equal to t_{End}.

The implementation of assignments within the schedule is based on the algebraic structures described in Subsection 2.1.1. The set of assignments that may be planned within the current schedule is denoted $\mathcal{A}$, with its various subsets denoted $\mathcal{A}_\circ$, $\mathcal{A}_{ij}$, and $\mathcal{A}_i$. The set of assignment types $\mathcal{T}_i$ categorizes the members of $\mathcal{A}_i$. This set notation is used in conjunction with the obj and class mappings that were introduced in Subsection 2.2.1.

For every assignment $l \in \mathcal{A}$, the planned duration is represented by $d^l \in \mathbb{R}^+_\circ$ and the planned completion time by $t^l \in \mathbb{R}$, which are both measured in hours. Additionally, the assignment type determinants,

$$\beta^l_{\text{Type}k} = \begin{cases} 1 & \text{if assignment } l \text{ is of type } k \\ 0 & \text{otherwise} \end{cases}$$

are defined for all $l \in \mathcal{A}$ and all $k \in \mathcal{T}_{\text{class}(l)}$. These binary quantities $\beta^l_{\text{Type}k}$ are the assignment type determinants. They are related to the categorical variables Type^l introduced in Chapter 2, such that $\beta^l_{\text{Type}k} = 1$ if and only if $\text{Type}^l = k$. If $\text{class}(l) = \text{PSC}$, or another state-machine

class, then $\beta^l_{\text{Type}k}$ may be called a transition-type determinant.

The set $\mathcal{A}_\circ$ contains the final assignments from the previous schedule, which are predetermined. Thus for each $l \in \mathcal{A}_\circ$, the quantities d^l, t^l and $\beta^l_{\text{Type}k}$ are parameters that describe the initial conditions of the system. It is understood that

$$t^l - d^l \leq t_{\text{Begin}}$$

for all $l \in \mathcal{A}_\circ$, because these assignments began in the previous schedule. Whether or not they extend into the current schedule depends on whether or not $t^l \geq t_{\text{Begin}}$, but either case is permitted. Additionally, the members of $\mathcal{A}_\circ$ have exactly one type,

$$\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^l_{\text{Type}k} = 1$$

for all $l \in \mathcal{A}_\circ$.

For the current assignments $\mathcal{A} \setminus \mathcal{A}_\circ$, the quantities d^l, t^l and $\beta^l_{\text{Type}k}$ are variables to be evaluated as part of the optimization. The current assignments must begin in the current schedule,

$$t^l - d^l \geq t_{\text{Begin}} \tag{4.1}$$

for all $\mathcal{A} \setminus \mathcal{A}_\circ$. Assignment l begins after its predecessor has been completed,

$$t^l - d^l \geq t^{l-} \tag{4.2}$$

for all $\mathcal{A} \setminus \mathcal{A}_\circ$, where $l- = (l_1, l_2, l_3 - 1)$ is the predecessor of $l = (l_1, l_2, l_3)$. Assignments may have at most one type,

$$\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^l_{\text{Type}k} \leq 1 \tag{4.3}$$

for all $\mathcal{A} \setminus \mathcal{A}_\circ$.

An assignment l which is not given any type, $\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^l_{\text{Type}k} = 0$, is undetermined. Using the categorical notation of Chapter 2, $\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^l_{\text{Type}k} = 0$ if and only if $\text{Type}^l = \text{Undetermined}$. In this case, the duration d^l is set to zero, as per

$$d^l \leq \overline{d} \sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^l_{\text{Type}k} \tag{4.4}$$

for all $\mathcal{A} \setminus \mathcal{A}_\circ$, where $\overline{d}$ is an upper bound on the duration of an assignment. It suffices to

take $\overline{d} = \bar{t} - t_{\text{Begin}}$, unless lower estimates are available (See Appendix B.5).

An assignment may not be determined in the current schedule unless the predecessor is determined,

$$\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l}_{\text{Type}k} \leq \sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l-}_{\text{Type}k} \tag{4.5}$$

for all $\mathcal{A} \setminus \mathcal{A}_{\circ}$. Thus, $\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l-}_{\text{Type}k} = 0$ implies $\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l}_{\text{Type}k} = 0$, so that l remains undetermined.

An assignment which begins in the current schedule can extend into the next schedule, but only until time $\bar{t}$,

$$t^{l-} \leq t_{\text{End}} + (\bar{t} - t_{\text{End}}) \left(1 - \sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l}_{\text{Type}k} \right) \tag{4.6}$$

for all $\mathcal{A} \setminus \mathcal{A}_{\circ}$. If there is an assignment $l-$ begins in the current schedule, and extends into the next schedule, then the successor l is undetermined. Thus if $t^{l-} \geq t_{\text{End}}$, then Equation 4.6 can only be satisfied if $\sum_{k \in \mathcal{T}_{\text{class}(l)}} \beta^{l}_{\text{Type}k} = 0$ so that l is indeed undetermined.

Equation 4.6 implicitly places the upper bound $\bar{t}$ on t^{l} for all $l \in \mathcal{A}$ except for when $l = (i, j, \overline{n}_{\text{asgn}i})$. To fully implement the desired upper bound,

$$t^{(i,j,\overline{n}_{\text{asgn}i})} \leq \bar{t} \tag{4.7}$$

for all $i \in \mathcal{C}$ and $j \in \{1, 2, \ldots, n_i\}$.

Equations 4.1-7 establish the general logic which enforces the ordering and categorization of the assignments. Following the treatment of Subsection 2.1.1, the assignment values of d^{l}, t^{l} and $\beta^{l}_{\text{Type}k}$ are sufficient to construct a Gantt Chart.

4.1.2 Dependencies

A PS system is composed of various objects that support the converting process. The interdependency of these objects is described by the sets $\mathcal{D}_{ik}$ which were introduced in Subsection 2.1.2.

For every class $i \in \mathcal{C}$, every assignment type $k \in \mathcal{T}_i$, and every dependency clause

$(i', k', n) \in \mathcal{D}_{ik}$, the following binary variables are defined,

$$\beta^{lk}_{\text{Supp}l'k'} = \begin{cases} 1 & \text{if assignment } l \text{ is of type } k, \text{ and is supported by } l' \text{ that is of type } k' \\ 0 & \text{otherwise} \end{cases}$$

for all $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$ and all $l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ$. These are the assignment support determinants.

The logic of each dependency clause is implemented through the following equality,

$$\sum_{l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ} \beta^{lk}_{\text{Supp}l'k'} = n\beta^{l}_{\text{Type}k} \tag{4.8}$$

for all $i \in \mathcal{C}$, $k \in \mathcal{T}_i$, $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$ and $(i', k', n) \in \mathcal{D}_{ik}$. Equation 4.8 prevents assignment l from being of type k, unless there are exactly n objects of class i' that have supporting assignments l' that are of type k'.

To ensure consistency between the support and type determinants,

$$2\beta^{lk}_{\text{Supp}l'k'} \leq \beta^{l}_{\text{Type}k} + \beta^{l'}_{\text{Type}k'} \tag{4.9}$$

for all $i \in \mathcal{C}$, $k \in \mathcal{T}_i$, $(i', k', n) \in \mathcal{D}_{ik}$, $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$ and $l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ$. If assignment l' is of a type k' in support of assignment l that is of type k, then $\beta^{lk}_{\text{Supp}l'k'} = 1$, and Equation 4.9 can only be satisfied if both l and l' have the corresponding assignment types, $\beta^{l}_{\text{Type}k} = \beta^{l'}_{\text{Type}k'} = 1$.

The following equality associates each supporting assignment with exactly one dependent assignment,

$$\beta^{l'}_{\text{Type}k'} = \sum_{l \in \mathcal{A}_i \setminus \mathcal{A}_\circ} \beta^{lk}_{\text{Supp}l'k'} \tag{4.10}$$

for all $i \in \mathcal{C}$, $k \in \mathcal{T}_i$, $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$, $(i', k', n) \in \mathcal{D}_{ik}$ and $l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ$. If assignment l' is of a type k' which supports type $k \in \mathcal{T}_i$, then $\beta^{l'}_{\text{Type}k'} = 1$ and Equation 4.10 ensures that there is exactly one assignment $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$ of type k that is supported by l'; otherwise, if l' is not of type k', then $\beta^{l'}_{\text{Type}k'} = 0$ and the equation ensures that there are no assignments of type k that are supported by l'.

For l to be supported by l', these two assignments must be simultaneous. This implies a coordination of the durations, $d^l = d^{l'}$, and of the completion times $t^l = t^{l'}$. Firstly for the

durations,

$$d^l \geq d^{l'} - \overline{d}\left(1 - \beta^{lk}_{\text{Suppl}'k'}\right) \tag{4.11}$$
$$d^l \leq d^{l'} + \overline{d}\left(1 - \beta^{lk}_{\text{Suppl}'k'}\right) \tag{4.12}$$

for all $i \in \mathcal{C}$, $k \in \mathcal{T}_i$, $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$, $(i', k', n) \in \mathcal{D}_{ik}$ and $l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ$. If l' is in support of l then the terms in the parentheses disappear; Equations 4.11-12 become $d^{l'} \leq d^l \leq d^{l'}$, respectively, which implies $d^l = d^{l'}$, as desired. Otherwise, if l' does not support l, then Equation 4.11 becomes $d^l \geq d^{l'} - \overline{d}$ which is automatically satisfied because $d^l \geq 0 \geq d^{l'} - \overline{d}$, and Equation 4.12 becomes $d^l \leq d^{l'} + \overline{d}$ which is automatically satisfied because $d^l \leq \overline{d} \leq d^{l'} + \overline{d}$.

To impose simultaneity for the completion times,

$$t^l \geq t^{l'} - \overline{t}\left(1 - \beta^{lk}_{\text{Suppl}'k'}\right) \tag{4.13}$$
$$t^l \leq t^{l'} + \overline{t}\left(1 - \beta^{lk}_{\text{Suppl}'k'}\right) \tag{4.14}$$

for all $i \in \mathcal{C}$, $k \in \mathcal{T}_i$, $l \in \mathcal{A}_i \setminus \mathcal{A}_\circ$, $(i', k', n) \in \mathcal{D}_{ik}$ and $l' \in \mathcal{A}_{i'} \setminus \mathcal{A}_\circ$. These two equations operate similarly to Equations 4.11-12.

Having incorporated the general features of a Gantt chart into the MILP, the remainder of the chapter focuses on the particularities of Peirce-Smith converters.

4.2 PS Converters as State-Machines

4.2.1 States and Transitions

The state of each Peirce-Smith converter is characterized by its mass and heat content, as well as the mechanistic aspects, e.g. whether or not a converter is ready to be charged. Under a state-machine model, the converters evolve throughout the set of transitions $\mathcal{A}_{\text{PSC}}$.

The state-machine implementation of the PSC class relies on the algebraic structures described in Chapter 3. This includes the sets of elements $\mathcal{E}$, product species $\mathcal{S}_{\text{Prod}}$, streams $\mathcal{Z}$, and flows $\mathcal{F}$, and their various subsets, as well as the srce mapping.

The following state quantities are defined in the MILP for all $l \in \mathcal{A}_{\text{PSC}}$.

- $m^l_{\text{Ret}k} \in \mathbb{R}^+_\circ$ is the mass of feed stream k retained in the bath of obj(l) at the end of transition l, for all $k \in \mathcal{Z}_{\text{NGFeed}}$.
- $m^l_{j\text{RetProd}} \in \mathbb{R}^+_\circ$ is the mass of species j within the retained product streams of obj(l) at

the end of transition l, for all $j \in \mathcal{S}_{\text{NGProd}}$.

- $h^l_{\text{Ret}} \in \mathbb{R}$ is the heat that is retained in the bath of obj(l) at the end transition l.
- $\beta^l_{\text{Type}k} \in \{0,1\}$ is the transition-type determinant described in Subsection 4.1.1, for all $k \in \mathcal{T}_{\text{PSC}}$.

For $l \in \mathcal{A}_{\text{PSC}} \bigcap \mathcal{A}_\circ$, these quantities are parameters that describe the initial state of the system. Otherwise, for $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$, they are state variables which are determined as part of the optimization. Throughout the current formulation, masses, such as $m^l_{\text{Ret}k}$ and $m^l_{j\text{RetProd}}$, are measured in tonnes. Heats, such as h^l_{Ret}, are measured in MJ, and with respect to the standard reference temperature 298.15 K. Incidentally, if obj(l) is empty at time t^l, then $m^l_{\text{Ret}k} = 0$ for all $k \in \mathcal{Z}_{\text{NGFeed}}$, $m^l_{j\text{RetProd}} = 0$ for all $j \in \mathcal{S}_{\text{Prod}}$, and $h^l_{\text{Ret}} = 0$.

At time t^l, the total mass of the bath in obj(l) can be computed as the sum of nongaseous feed and product streams, $\left(\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k} + \sum_{j\in\mathcal{S}_{\text{Prod}}} m^l_{j\text{RetProd}}\right)$. It is appropriate that the product streams should be expressed in a speciated form, which allows the composition of the product streams to be deduced. In contrast, the feed streams are better expressed in the simpler, unspeciated form. In any case, the speciation data may not even be available for all of the feed streams, as described in Subsection 3.3.2.

The following transition variables are defined in the MILP for all current converter transitions $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$.

- $d^l_i \in \mathbb{R}^+_\circ$ is the duration of segment i of transition l, for all $i \in \{0, 1, 2, \ldots, 7\}$.
- $v^{jl} \in \mathbb{R}^+_\circ$ is the volume of srce(j) delivered during transition l via mechanism j, to or from obj(l), for all $j \in \mathcal{F}_{\text{NG}}$.
- $u^{jl} \in \mathbb{Z}^+_\circ$ is the number of delivery units used during transition l to carry srce(j) via mechanism j, to or from obj(l), for all $j \in \mathcal{F}^{\text{MSM}}$.
- $\beta^l_{\text{Type}k} \in \{0,1\}$ is the transition-type determinant described in Subsection 4.1.1, for all $k \in \mathcal{T}_{\text{PSC}}$.

The transition-type determinants $\beta^l_{\text{Type}k}$ have a double role, characterizing the mechanistic preparedness, as well as the transitions. They are thus regarded as both state and transition variables.

Converter transitions are related to the general Gantt structure,

$$d^l = \sum_{i=1}^{7} d^l_i \tag{4.15}$$

for all $l \in \mathcal{A}_{\rm PSC} \setminus \mathcal{A}_\circ$, which considers segments one through seven. The "zeroth" segment refers to the idle time which occurs between transitions $l-$ and l, such that

$$d_\circ^l = t^l - d^l - t^{l-} \tag{4.16}$$

for all $l \in \mathcal{A}_{\rm PSC} \setminus \mathcal{A}_\circ$.

To ensure the correct interpretation of $\mathcal{T}_{\rm PSC,Empty}$,

$$\sum_{k \in \mathcal{Z}_{\rm NGFeed}} \frac{m^l_{{\rm Ret}k}}{\rho_k} + \sum_{j \in \mathcal{S}_{\rm NGProd}} \frac{m^l_{j{\rm RetProd}}}{\rho_j} \leq \overline{v}^{{\rm obj}(l)} \left(1 - \sum_{k \in \mathcal{T}_{\rm PSC,Empty}} \beta^l_{{\rm Type}k} \right) \tag{4.17}$$

for all $l \in \mathcal{A}_{\rm PSC} \setminus \mathcal{A}_\circ$, where ρ_k and ρ_j are the density of feed stream k, and of product species j, respectively. Equation 4.17 ensures that no bath is retained when the obj(l) is emptied. Similarly,

$$\begin{aligned} h^l_{\rm Ret} &\geq \underline{h}^{{\rm obj}(l)} \left(1 - \sum_{k \in \mathcal{T}_{\rm PSC,Empty}} \beta^l_{{\rm Type}k} \right) && (4.18) \\ &\leq \overline{h}^{{\rm obj}(l)} \left(1 - \sum_{k \in \mathcal{T}_{\rm PSC,Empty}} \beta^l_{{\rm Type}k} \right) && (4.19) \end{aligned}$$

for all $l \in \mathcal{A}_{\rm PSC} \setminus \mathcal{A}_\circ$, so that no heat is retained when obj(l) is empty or is being emptied.

Equations 4.17-19 utilize volume and heat bounds $\overline{v}^{{\rm obj}(l)}$, $\underline{h}^{{\rm obj}(l)}$ and $\overline{h}^{{\rm obj}(l)}$, which are associate to a converter objects, obj(l) = (PSC,j) for $j \in \{1, 2, \ldots, n_{\rm PSC}\}$. The volume bound $\overline{v}^{{\rm PSC}j}$ is the maximum bath volume that is observed or allowed in converter j, usually given by the vessel geometry. The heat bounds are given by

$$\begin{aligned} \underline{h}^{{\rm PSC}j} &= \min \left(\min_{k \in \mathcal{Z}_{\rm NGFeed}} [w_{{\rm H}k}(\underline{T}^{{\rm PSC}j})] \rho_k \overline{v}^{{\rm PSC}j}, \min_{j' \in \mathcal{S}_{\rm NGProd}} [w_{{\rm H}j'}(\underline{T}^{{\rm PSC}j})] \rho_j \overline{v}^{{\rm PSC}j}, 0 \right) \\ \overline{h}^{{\rm PSC}j} &= \max \left(\max_{k \in \mathcal{Z}_{\rm NGFeed}} [w_{{\rm H}k}(\overline{T}^{{\rm PSC}j})] \rho_k \overline{v}^{{\rm PSC}j}, \max_{j' \in \mathcal{S}_{\rm NGProd}} [w_{{\rm H}j'}(\overline{T}^{{\rm PSC}j})] \rho_j \overline{v}^{{\rm PSC}j}, 0 \right) \end{aligned}$$

in which $\underline{T}^{{\rm PSC}j}$ and $\overline{T}^{{\rm PSC}j}$, are the minimum and maximum bath temperatures that can be observed or allowed in converter j. The specific heats $w_{{\rm H}k}(\underline{T}^{{\rm PSC}j})$,$w_{{\rm H}j'}(\underline{T}^{{\rm PSC}j})$, $w_{{\rm H}k}(\overline{T}^{{\rm PSC}j})$, and $w_{{\rm H}j'}(\overline{T}^{{\rm PSC}j})$, are computed as described Subsection 3.2.3.

This formulation considers the theoretical possibility that the nongaseous streams all have

positive specific heat contents, so that $\underline{h}^{\mathrm{PSC}j} = 0$ is obtained when the converter is empty; this situation does not actually occur in practice, and typical values of $\underline{h}^{\mathrm{PSC}j}$ range between -10^8MJ and -10^6MJ. Similarly, there is the possibility of $\overline{h}^{\mathrm{PSC}j} = 0$, in case the nongaseous streams all have negative specific heat contents. In practice, $\overline{h}^{\mathrm{PSC}j}$ is typically between 10^6MJ and 10^8MJ.

Usually it is sufficient to set $\underline{T}^{\mathrm{PSC}j}$ equal to either the blast temperature, or the ambient smelter temperature, whichever is lower. On the other hand, $\overline{T}^{\mathrm{PSC}j}$ can be set tactically, depending on the thermochemical behaviour of the converting process, and respecting the tolerance of the refractory lining. Sections 5.3-4 use $\underline{T}^{\mathrm{PSC}j} = 30°\mathrm{C}$ and $\overline{T}^{\mathrm{PSC}j} = 1250°\mathrm{C}$ to perform sample calculations.

Having already discussed $\mathcal{T}_{\mathrm{PSC,Empty}}$, the notion of mechanistic preparedness is completed by,

$$\beta^l_{\mathrm{Type}k} \leq \sum_{k' \in \mathcal{T}^-_{\mathrm{PSC}k}} \beta^{l-}_{\mathrm{Type}k'} \tag{4.20}$$

for all $k \in \mathcal{T}_{\mathrm{PSC}}$ and $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ$, in which $\mathcal{T}^-_{\mathrm{PSC}k} \subset \mathcal{T}_{\mathrm{PSC}}$ are the transition types that may immediately precede type k.

The transition variables are bounded according to the transition types. For all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ$,

$$\begin{aligned} d^l_i &\geq \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \underline{d}^k_i \beta^l_{\mathrm{Type}k} && (4.21)\\ &\leq \overline{d}_i - \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \left(\overline{d}_i - \overline{d}^k_i\right) \beta^l_{\mathrm{Type}k} && (4.22)\\ &\leq \overline{d}_i \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \beta^l_{\mathrm{Type}k} && (4.23) \end{aligned}$$

for all $i \in \{0,1,\ldots,7\}$, and

$$\begin{aligned} v^{jl} &\geq \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \underline{v}^{jk} \beta^l_{\mathrm{Type}k} && (4.24)\\ &\leq \overline{v}^j - \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \left(\overline{v}^j - \overline{v}^{jk}\right) \beta^l_{\mathrm{Type}k} && (4.25)\\ &\leq \overline{v}^j \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \beta^l_{\mathrm{Type}k} && (4.26) \end{aligned}$$

for all $j \in \mathcal{F}_{\text{NG}}$, and

$$
\begin{aligned}
u^{jl} &\geq \sum_{k \in \mathcal{T}_{\text{PSC}}} \underline{u}^{jk} \beta^{l}_{\text{Type}k} && (4.27)\\
&\leq \overline{u}^{j} - \sum_{k \in \mathcal{T}_{\text{PSC}}} \left(\overline{u}^{j} - \overline{u}^{jk} \right) \beta^{l}_{\text{Type}k} && (4.28)\\
&\leq \overline{u}^{j} \sum_{k \in \mathcal{T}_{\text{PSC}}} \beta^{l}_{\text{Type}k} && (4.29)
\end{aligned}
$$

for all $j \in \mathcal{F}^{\text{MSM}}$, in which

$$
\begin{aligned}
\overline{d}_i &= \max_{k \in \mathcal{T}_{\text{PSC}}} \overline{d}^{k}_{i} \\
\overline{v}^{j} &= \max_{k \in \mathcal{T}_{\text{PSC}}} \overline{v}^{jk} \\
\overline{u}^{j} &= \max_{k \in \mathcal{T}_{\text{PSC}}} \overline{u}^{jk}
\end{aligned}
$$

Equations 4.21-22 impose duration limits, e.g. $(\underline{d}^{k}_{\circ}, \overline{d}^{k}_{\circ}) = (1, 3)$ implies that a converter must remain idle between 1 and 3 hours before a transition of type k may begin. Equations 4.24-25 have the same role for the flow volumes, and Equations 4.27-28 for the delivery units. Equations 4.23, 4.26 and 4.29 ensure that the transition variables are zero when the transition is undetermined.

Appropriate default values can be obtained for $\overline{d}^{k}_{i}$ by considering general bounds within the Gantt structure. For instance,

$$
\overline{d}^{k}_{\circ} = \begin{cases} t_{\text{Begin}} - t_{\text{End}} & \text{if transitions of type } k \text{ may be preceded by idle time} \\ 0 & \text{otherwise} \end{cases}
$$

would typically determine $\overline{d}^{k}_{\circ}$ when no lower estimates are forthcoming. The remaining segments $i \in \{1, \ldots, 7\}$ can be determined according to

$$
\overline{d}^{k}_{i} = \begin{cases} \overline{d} & \text{if segment } i \text{ is included in transitions of type } k \\ 0 & \text{otherwise} \end{cases}
$$

unless better (lower) estimates can be found.

It is reasonable for the nongaseous flows $j \in \mathcal{F}_{\text{NG}}$ that

$$
\overline{v}^{jk} \leq \max_{j' \in \{1, 2, n_{\text{PSC}}\}} \overline{v}^{\text{PSC}j'}
$$

where $\overline{v}^{\mathrm{PSC}j'}$ is once again the maximum bath volume that can be contained in the converter j'.

The flow volumes and units are related,

$$v^{jl} = v_u^j u^{jl} \tag{4.30}$$

for the modulated flows $j \in \mathcal{F}^{\mathrm{M}}$,

$$v^{jl} \leq \overline{v}_u^j u^{jl} \tag{4.31}$$

for the semi-modulated flows $j \in \mathcal{F}^{\mathrm{SM}}$, for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$. As discussed in Subsection 3.5.2, v_u^j and $\overline{v}_u^j$ are the volumetric capacities of modulated delivery units, and of semi-modulated delivery units, respectively. These constants are used to deduce appropriate default values for $\overline{u}^{jk}$, as

$$\overline{u}^{jk} = \begin{cases} \lfloor \frac{\max_{j' \in \{1,2,n_{\mathrm{PSC}}\}} \overline{v}^{\mathrm{PSC}j'}}{v_u^j} \rfloor & \text{if mechanism } j \text{ functions during transitions of type } k \\ 0 & \text{otherwise} \end{cases}$$

for $j \in \mathcal{F}^{\mathrm{M}}$, and

$$\overline{u}^{jk} = \begin{cases} \lceil \frac{\max_{j' \in \{1,2,n_{\mathrm{PSC}}\}} \overline{v}^{\mathrm{PSC}j'}}{\overline{v}_u^j} \rceil & \text{if mechanism } j \text{ functions during transitions of type } k \\ 0 & \text{otherwise} \end{cases}$$

As described in Appendix B.5, it is be preferable to use lower values for $\overline{u}^{jk}$ if such values could be obtained without deteriorating the discrete feasibility region.

It is common for a semi-modulated flows to have a modulated counterpart. SlagPartialLadle $\in \mathcal{F}_{\mathrm{DCh}}^{\mathrm{SM}}$ may be complemented by SlagFullLadle $\in \mathcal{F}_{\mathrm{DCh}}^{\mathrm{M}}$, which both involve the same ladles $v_u^{\mathrm{SlagFullLadle}} = \overline{v}_u^{\mathrm{SlagPartialLadle}}$, and which both function during Skim transitions. All other factors being equal,

$$\begin{aligned} \overline{u}^{\mathrm{SlagFullLadle,Skim}} &= \lceil \frac{\max_{j' \in \{1,2,n_{\mathrm{PSC}}\}} \overline{v}^{\mathrm{PSC}j'}}{v_u^{\mathrm{SlagFullLadle}}} \rceil - 1 \\ \overline{u}^{\mathrm{SlagPartialLadle,Skim}} &= 1 \end{aligned}$$

Thus a skimming operation may completely fill several ladles, and only the final ladle would be partially filled.

The state variables ($m_{\mathrm{Ret}k}^l$,$m_{j\mathrm{Ret}}^l$, h_{Ret}^l, $\beta_{\mathrm{Type}k}^l$) and transition variables (d_i^l,v^{jl}, u^{jl}, $\beta_{\mathrm{Type}k}^l$)

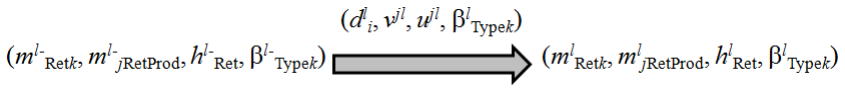


Figure 4.1: Schematic representation of a converter transition

have a general interaction, depicted by Figure 4.1. The variables describing transition l are used to update the preceding state variables ($l-$), and thus to obtain the current state variables (l). This updating process is called the forward state computation, discussed in Section 4.4.

4.2.2 Converting Actions

Following the treatment of Subsection 2.2.2, the duration segments have the interpretation that $d^l_{\text{Ch}} = d^l_2$, $d^l_{\text{Blow}} = d^l_4$, and $d^l_{\text{DCh}} = d^l_6$, referring to the charge, blow, and discharge actions, respectively.

A transition type k that does not include any charging time is such that $\underline{d}^k_2 = \overline{d}^k_2 = 0$, as per Equations 4.21-22. Likewise, $\underline{d}^k_4 = \overline{d}^k_4 = 0$ implies no blowing time, and $\underline{d}^k_6 = \overline{d}^k_6 = 0$ implies no discharging time. The elimination of segments should follow the convention illustrated by Figures 2.7 and 2.8, to use the lowest indices possible, while respecting $d^l_2 = d^l_{\text{Ch}}$, $d^l_4 = d^l_{\text{Blow}}$, and $d^l_6 = d^l_{\text{DCh}}$. This convention is preserved through the following logic,

$$\left(\overline{d}^k_{(2i+1)} = 0\right) \text{ implies } \left(\overline{d}^k_{(2i)} = \underline{d}^k_{(2i)} = \underline{d}^k_{(2i+1)} = 0\right)$$

for $i \in \{1, 2, 3\}$. In future discussion, the action labeling is applied to the lower and upper bounds, so that $\underline{d}^k_{\text{Ch}} = \underline{d}^k_2$, $\overline{d}^k_{\text{Ch}} = \overline{d}^k_2$, $\underline{d}^k_{\text{Blow}} = \underline{d}^k_4$, $\overline{d}^k_{\text{Blow}} = \overline{d}^k_4$, $\underline{d}^k_{\text{DCh}} = \underline{d}^k_6$, and $\overline{d}^k_{\text{DCh}} = \overline{d}^k_6$.

The action durations are related to the flow mechanisms and transition type. Firstly, the charge duration $d^l_{\text{Ch}} = d^l_2$ is taken to be a linear combination of the charing volumes and units, plus a constant term that is associated to the transition type,

$$d^l_{\text{Ch}} = \sum_{j \in \mathcal{F}_{\text{Ch}}} d^j_v v^{jl} + \sum_{j \in \mathcal{F}^{\text{MSM}}_{\text{Ch}}} d^j_u u^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} d_{\text{Ch}k} \beta^l_{\text{Type}k} \tag{4.32}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Similarly,

$$d^l_{\text{DCh}} = \sum_{j \in \mathcal{F}_{\text{DCh}}} d^j_v v^{jl} + \sum_{j \in \mathcal{F}^{\text{MSM}}_{\text{DCh}}} d^j_u u^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} d_{\text{DCh}k} \beta^l_{\text{Type}k} \tag{4.33}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. During the blowing action, the nongaseous feed is limited,

$$d^l_{\text{Blow}} \geq \sum_{j \in \mathcal{F}_{\text{NGBlow}}} d^j_v v^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} d_{\text{Blow}k} \beta^l_{\text{Type}k} \tag{4.34}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. The flow duration constants d^j_v are defined for all nongaseous flows $j \in \mathcal{F}_{\text{NG}}$, and d^j_u for all modulated and semi-modulated flows $j \in \mathcal{F}^{\text{MSM}}$. Additionally, $d_{\text{Ch}k}$, $d_{\text{Blow}k}$ and $d_{\text{DCh}k}$ are defined for all $k \in \mathcal{T}_{\text{PSC}}$.

Equations 4.32-34 ensure consistency among the transition variables, and provide the appropriate interpretation of the transition segments. This interpretation is important when establishing the transition feasibility conditions, including composition, volumetric and thermal constraints, as described in Section 4.5.

4.3 Intermediate Computations

4.3.1 Intermediate Variables

Subsection 4.2.1 presented MILP variables which uniquely determine converter states and transitions. The following intermediate variables are used to develop the forward state computation (Section 4.4) and to test the feasibility of transitions (Section 4.5). These variables are listed in the order in which they are treated.

- $m^l_k \in \mathbb{R}^+_{\circ}$ is the mass of feed stream k that participates in transition l, for all $k \in \mathcal{Z}_{\text{NGFeed}}$.
- $m^l_{i\text{Prod}} \in \mathbb{R}^+_{\circ}$ is the mass of element i within the product streams of transition l, for all $i \in \mathcal{E}$.
- $h^l_{\text{Ch}} \in \mathbb{R}$ is the heat that is introduced into obj(l) as part of the charge streams of transition l.
- $h^l_{\text{NGBlow}} \in \mathbb{R}$ is the heat that is introduced into obj(l) as part of the nongaseous blow streams of transition l.
- $m^l_{i\text{Blast}} \in \mathbb{R}^+_{\circ}$ is the mass of element i that is blown into the melt as part of the blast of transition l, for all $i \in \mathcal{E}$.
- $m^l_{j\text{Prod}} \in \mathbb{R}^+_{\circ}$ is the mass of species j within the product streams of transition l, for all $j \in \mathcal{S}_{\text{Prod}}$.
- $h^l_{\text{Blast}} \in \mathbb{R}$ is the heat that is blown into the melt as part of the blast of transition l.

- $h^l_{\text{Offgas}} \in \mathbb{R}$ is the heat that is convected out of obj(l) as part of the offgas stream of transition l.
- $h^l_{\text{DCh}} \in \mathbb{R}$ is the heat that is removed from obj(l) as part of the discharge streams of transition l.
- $h^l_{\text{Env}i} \in \mathbb{R}^+_\circ$ is the heat that is lost from the bath of obj(l) to the environment during segment i of transition l, for all $i \in \{0, 1, \ldots, 7\}$.

These quantities are variables within the MILP, for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$.

From the definitions, it follows that

$$m^l_{\text{Ret}k} \leq m^l_k$$

since there cannot be more of stream k retained than there is available, for all $k \in \mathcal{Z}_{\text{NGFeed}}$. Similarly,

$$m^l_{j\text{RetProd}} \leq m^l_{j\text{Prod}}$$

These two inequalities are not explicitly included in the MILP, since they are implied by other constraints, described below.

A distinction is made between essential and nonessential intermediate variables. Essential intermediate variables are those which cannot be substituted by a linear combination of the preceding state variables ($l-$) and current transition variables (l), but are required either for the forward state computation, or for testing the feasibility of the proposed transition.

The feed masses m^l_k are nonessential since they can be computed through

$$m^l_k = m^{l-}_{\text{Ret}k} + \rho_k \sum_{j \in \mathcal{F}_k} v^{jl} \tag{4.35}$$

for all $k \in \mathcal{Z}_{\text{NGFeed}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Equation 4.35 includes the retained contribution from the previous transition, plus the newly added contribution from the current transition.

The product elemental masses are also nonessential,

$$m^l_{i\text{Prod}} = \sum_{j \in \mathcal{S}_{\text{NGProd}}} w_{ij} m^{l-}_{j\text{RetProd}} + \sum_{k \in \mathcal{Z}_{\text{NGFeed}}} w_{ik} \left(m^l_k - m^l_{\text{Ret}k} \right) + m^l_{i\text{Blast}} \tag{4.36}$$

for all $i \in \mathcal{E}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$, where w_{ij} and w_{ik} are the weight fractions of i in species j, and in stream k, respectively. On the righthand side of Equation 4.36, the first summation

is the mass of i that is retained in the product streams from the previous transition. This is followed by contributions from the nongaseous feed, but only the unretained component is reacted. The final term is the contribution from the blast.

The charge and nongaseous blow heats are computed according to

$$h^l_{\mathrm{Ch}} = \sum_{j \in \mathcal{F}_{\mathrm{Ch}}} w^j_{\mathrm{H,srce}(j)} \rho_{\mathrm{srce}(j)} v^{jl} \tag{4.37}$$

and

$$h^l_{\mathrm{NGBlow}} = \sum_{j \in \mathcal{F}_{\mathrm{NGBlow}}} w^j_{\mathrm{H,srce}(j)} \rho_{\mathrm{srce}(j)} v^{jl} \tag{4.38}$$

respectively, for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$, where $w^j_{\mathrm{H,srce}(j)} = w_{\mathrm{H,srce}(j)}(T^j)$ is the specific heat content of srce(j) evaluated at delivery temperature T^j.

These nonessential intermediate variables are implemented mainly to enhance the readability of Sections 4.4 and 4.5. However, it is possible to eliminate these variables from the optimization through substitution, and then to compute them *a posteriori*, after the optimization is complete.

Table 4.1 classifies all of the variables used by the MILP to describe the evolution of a converter. The remainder of this section describes the computation of the essential intermediate variables.

Table 4.1: Classification of the converter variables included in the current MILP implementation

Classification	Variables					
State Variables	$m^l_{\mathrm{Ret}k}$,	$m^l_{j\mathrm{RetProd}}$,	h^l_{Ret},	$\beta^l_{\mathrm{Type}k}$		
Transition Variables	d^l_i,	v^{jl},	u^{jl},	$\beta^l_{\mathrm{Type}k}$		
Nonessential Intermediate Variables	m^l_k,	$m^l_{i\mathrm{Prod}}$,	h^l_{Ch},	h^l_{NGBlow}		
Essential Intermediate Variables	$m^l_{i\mathrm{Blast}}$,	$m^l_{j\mathrm{Prod}}$,	h^l_{Blast},	h^l_{Offgas},	h^l_{DCh},	$h^l_{\mathrm{Env}i}$

$\beta^l_{\mathrm{Type}k}$ qualify as both state and transition variables

4.3.2 Blast Elemental Masses

As described in Subsection 3.3.3, $m^l_{i\text{Blast}}$ is proportional to the blowing time d^l_{Blow}. However, the proportionality constant depends on the transition type. There may be different oxygen enrichment between the Slag-Blow and the Copper-Blow transitions, for example.

The blowing masses can be computed as

$$m^l_{i\text{Blast}} = \begin{cases} \dot{m}^k_{i\text{Blast}} d^l_{\text{Blow}} & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\text{PSC}} \\ 0 & \text{otherwise} \end{cases}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$ and $i \in \mathcal{E}$. Subsection 3.3.3 demonstrates how to compute the proportionality constants $\dot{m}^l_{\text{O,Blast}}$ and $\dot{m}^l_{\text{N,Blast}}$, from the oxygen enrichment $\phi^k_{\text{O}_2}$ and blast rate $\dot{v}^{\text{Norm}k}_{\text{Blast}}$, while $\dot{m}^l_{i\text{Blast}} = 0$ for all $i \in \mathcal{E} \setminus \{\text{O,N}\}$. Nonetheless, $\dot{m}^k_{i\text{Blast}}$ can be arbitrarily set to zero when $\overline{d}^k_{\text{Blow}} = 0$, in which transition type k does not allow blowing anyways.

The computation of $m^l_{i\text{Blast}}$ is implemented within the MILP using the following inequalities,

$$\begin{aligned} m^l_{i\text{Blast}} &\geq \dot{m}^k_{i\text{Blast}} d^l_{\text{Blow}} - \overline{m}_{i\text{Blast}}(1 - \beta^l_{\text{Type}k}) && (4.39) \\ &\leq \dot{m}^k_{i\text{Blast}} d^l_{\text{Blow}} + \overline{m}_{i\text{Blast}}(1 - \beta^l_{\text{Type}k}) && (4.40) \end{aligned}$$

for all $i \in \mathcal{E}$, all $k \in \mathcal{T}_{\text{PSC}}$, and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. The upper bound $\overline{m}_{i\text{Blast}}$ is taken as

$$\overline{m}_{i\text{Blast}} = \max_{k \in \mathcal{T}_{\text{PSC}}} \left(\dot{m}^k_{i\text{Blast}} \overline{d}^k_{\text{Blow}} \right)$$

Equations 4.39-40 are slack when $\beta^l_{\text{Type}k} = 0$, otherwise they force the desired proportionality $m^l_{i\text{Blast}} = \dot{m}^k_{i\text{Blast}} d^l_{\text{Blow}}$.

If transition l is undetermined, then $m^l_{i\text{Blast}}$ should be set to zero,

$$m^l_{i\text{Blast}} \leq \overline{m}_{i\text{Blast}} \sum_{k \in \mathcal{T}_{\text{PSC}}} \beta^l_{\text{Type}k} \qquad (4.41)$$

for all $i \in \mathcal{E}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Without Equation 4.41, the computed values of $m^l_{i\text{Blast}}$ would be erroneous for all of the undetermined transitions.

The blast masses $m^l_{i\text{Blast}}$ are essential intermediate variables due to the discrete effects imposed by the transition types. More generally, essential intermediate variables are formed as a clash between discrete and continuous effects.

4.3.3 Product Species Masses

As described in Section 3.4, the product species masses $\{m^l_{j\text{Prod}} |\ j \in \mathcal{S}_{\text{Prod}}\}$ depend on the reaction regime. They also depend on two performance indicators, the oxygen efficiency e_{O} and the ferroslag ratio r_{FS}.

The reaction regimes are implemented using the following binary variables, defined for all regimes $k \in \mathcal{R}$, and all converter transitions $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$,

$$\beta^l_{\text{Rg}k} = \begin{cases} 1 & \text{if the bath of obj}(l) \text{ is in regime } k \text{ at the end of transition } l \\ 0 & \text{otherwise} \end{cases}$$

These are called regime determinants. They may be regarded as essential intermediate variables, but perhaps they may be better described as "internal" variables, since their purpose is internal to the intermediate computations.

As described in Subsection 3.4.1, there is a subset of species $\mathcal{S}_{\text{RgProd}} \subset \mathcal{S}_{\text{NGProd}}$, whose stability depends on the reaction regime. This conditionality is implemented with

$$m^l_{j\text{Prod}} \leq \overline{m}_{j\text{Prod}} \sum_{k \in \mathcal{R}_j} \beta^l_{\text{Rg}k} \tag{4.42}$$

where $\mathcal{R}_j$ is the set of regimes that support species j, for all $j \in \mathcal{S}_{\text{RgProd}}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. The upper bound $\overline{m}_{j\text{Prod}}$ can be computed for all nongaseous product species $j \in \mathcal{S}_{\text{NGProd}} \supset \mathcal{S}_{\text{RgProd}}$,

$$\overline{m}_{j\text{Prod}} = \rho_j \max_{j' \in \{1,\ldots,n_{\text{PSC}}\}} \overline{v}^{\text{PSC}j'}$$

According to Equation 4.42, $m^l_{j\text{Prod}} = 0$ whenever l finishes in a regime for which j does not occur.

The product speciation balance is given by

$$\sum_{j \in \mathcal{S}_{\text{Prod}}} w_{ij} m^l_{j\text{Prod}} = m^l_{i\text{Prod}} \tag{4.43}$$

for all $i \in \mathcal{E}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. The righthand side of Equation 4.43 corresponds to the elemental masses given by Equation 4.36. Thus, the elemental masses are distributed into species masses, although some of the species are eliminated through Equation 4.42.

The regime determinants are subject to the following constraint that is explained below,

$$\sum_{k\in\mathcal{R}} \beta^l_{\mathrm{Rg}k} = |\mathcal{R}| - (|\mathcal{R}| - 1) \sum_{k\in\mathcal{T}_{\mathrm{PSC,Prod}}} \beta^l_{\mathrm{Type}k} \tag{4.44}$$

for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$, where $\mathcal{T}_{\mathrm{PSC,Prod}} \subset \mathcal{T}_{\mathrm{PSC}}$ is the set of transition types in which products streams are present.

The subset $\mathcal{T}_{\mathrm{PSC,Prod}}$ had not been introduced in Chapter 2, since it is an artifact of the MILP implementation, rather than an inherent feature of the PSC Problem. Earlier versions of the MILP did not consider $\mathcal{T}_{\mathrm{PSC,Prod}}$, and used the following equality instead of Equation 4.44,

$$\sum_{k\in\mathcal{R}} \beta^l_{\mathrm{Rg}k} = 1$$

so that every converter transition would be assigned one reaction type. This approach was theoretically sound, except that numerical instabilities occurred when $m^l_{i\mathrm{Prod}} \approx 0$ for all $i \in \mathcal{E}$, causing a conflict in the mass balance (Equation 4.43). Indeed, the reaction regime is not well-defined when there are no product streams. To eliminate the instability, Equation 4.44 has been formulated such that Equation 4.42 is slack for all $j \in \mathcal{S}_{\mathrm{RgProd}}$ whenever l is of a type $k \notin \mathcal{T}_{\mathrm{PSC,Prod}}$ or is undetermined.

The interpretation of $\mathcal{T}_{\mathrm{PSC,Prod}}$ should be respected, at least to within an acceptable numerical tolerance,

$$m^l_{j\mathrm{Prod}} \leq \overline{m}_{j\mathrm{Prod}} - (\overline{m}_{j\mathrm{Prod}} - \epsilon_{m\mathrm{Prod}}) \left(1 - \sum_{k\in\mathcal{T}_{\mathrm{PSC,Prod}}} \beta^l_{\mathrm{Type}k}\right) \tag{4.45}$$

for all $j \in \mathcal{S}_{\mathrm{Prod}}$ and $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$, where $\epsilon_{m\mathrm{Prod}}$ can be set to 0.0001 tonnes.

Following the discussion of Subsection 3.4.2, the solution of Equations 4.42-44 for the product species $\{m^l_{j\mathrm{Prod}} |\ j \in \mathcal{S}_{\mathrm{Prod}}\}$ depends on two performance parameters. Firstly, the oxygen enrichment is given by

$$e_{\mathrm{O}} = 1 - \left(\frac{m^l_{\mathrm{O_2,Prod}}}{m^l_{\mathrm{O,Blast}}}\right)$$

expressed as a linear constraint,

$$m^l_{\mathrm{O_2,Prod}} = (1 - e_{\mathrm{O}}) m^l_{\mathrm{O,Blast}} \tag{4.46}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. Secondly, the ferroslag ratio is given by

$$r_{\text{FS}} = \left(\frac{2M_{\text{FeO}}}{M_{\text{Fe}_2\text{SiO}_4}} \right) \left(\frac{m^l_{\text{Fe}_2\text{SiO}_4,\text{Prod}}}{m^l_{\text{Fe}_3\text{O}_4,\text{Prod}}} \right)$$

in which M_j is the molar mass of j, from which it follows that

$$r_{\text{FS}} m^l_{\text{Fe}_3\text{O}_4,\text{Prod}} = \left(\frac{2M_{\text{FeO}}}{M_{\text{Fe}_2\text{SiO}_4,\text{Prod}}} \right) m^l_{\text{Fe}_2\text{SiO}_4,\text{Prod}} \tag{4.47}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$.

If e_{O} and r_{FS} are specified parameters, then Equations 4.42-47 are a means to solve for $\{m^l_{j\text{Prod}} |\ j \in \mathcal{S}_{\text{Prod}}\}$, which is theoretically and numerically sound. Moreover, these equations are equally valid for the General Copper-Nickel PSC Formulation as for the Simplified Copper PSC Formulation.

4.3.4 Blast Heat

The blast heat h^l_{Blast} is closely related to the blast masses, as discussed in Subsection 3.3.3. Again, this quantity is proportional to the blowing duration d^l_{Blow}, with the proportionality constant being determined by the transition type.

Thus, the blast heat is computed according to

$$h^l_{\text{Blast}} = \begin{cases} \dot{h}^k_{\text{Blast}} d^l_{\text{Blow}} & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\text{PSC}} \\ 0 & \text{otherwise} \end{cases}$$

in which the proportionality constant $\dot{h}^k_{\text{Blast}}$ is computed as described in Subsection 3.3.3, using the specific heat of the blast species $w^{\text{Blast}k}_{\text{H}j} = w_{\text{H}j}(T^{\text{Blast}k})$. As before, $\dot{h}^k_{\text{Blast}}$ can be arbitrarily set to zero when $\overline{d}^k_{\text{Blow}} = 0$.

The blast heat h^l_{Blast} is implemented into the MILP in a manner similar to the blowing masses (Equations 4.39-40). However, the blowing heat is complicated by the fact that it can be either positive or negative. Using the standard reference temperature, 25°C, and the assumption that the blast is a mixture of O_2 and N_2, h^l_{Blast} can be positive or negative, depending on whether or not the blast is hotter or colder than 25°C. Either case is possible, depending on the air compression technology which provides the blast.

The following inequalities are slightly more complicated than Equations 4.39-40,

$$h^l_{\text{Blast}} \geq \dot{h}^k_{\text{Blast}} d^l_{\text{Blow}} - \left(\overline{h}_{\text{Blast}} - \underline{h}_{\text{Blast}}\right)\left(1 - \beta^l_{\text{Type}k}\right) \tag{4.48}$$

$$h^l_{\text{Blast}} \leq \dot{h}^k_{\text{Blast}} d^l_{\text{Blow}} + \left(\overline{h}_{\text{Blast}} - \underline{h}_{\text{Blast}}\right)\left(1 - \beta^l_{\text{Type}k}\right) \tag{4.49}$$

for all $k \in \mathcal{T}_{\text{PSC}}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. The bounds are given by

$$\underline{h}_{\text{Blast}} = \min\left(\min_{k \in \mathcal{T}_{\text{PSC}}} \dot{h}^k_{\text{Blast}} \overline{d}^k_{\text{Blow}}, 0\right)$$

$$\overline{h}_{\text{Blast}} = \max\left(\max_{k \in \mathcal{T}_{\text{PSC}}} \dot{h}^k_{\text{Blast}} \overline{d}^k_{\text{Blow}}, 0\right)$$

These bounds are constructed such that Equations 4.48-49 are slack when $\beta^l_{\text{Type}k} = 0$. Otherwise, when $\beta^l_{\text{Type}k} = 1$, it follows that $h^l_{\text{Blast}} = \dot{h}^k_{\text{Blast}} d^l_{\text{Blow}}$, as desired.

Undetermined transitions are handled through the following inequalities,

$$h^l_{\text{Blast}} \geq \underline{h}_{\text{Blast}} \sum_{k \in \mathcal{T}_{\text{PSC}}} \beta^l_{\text{Type}k} \tag{4.50}$$

$$h^l_{\text{Blast}} \leq \overline{h}_{\text{Blast}} \sum_{k \in \mathcal{T}_{\text{PSC}}} \beta^l_{\text{Type}k} \tag{4.51}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. When l is undetermined, Equations 4.50-51 force $h^l_{\text{Blast}} = 0$ rather than allowing erroneous values, and are hence comparable Equation 4.41.

The computation of blast heat is fairly elementary, and is well supported by the treatment of Section 3.3.3 and the MILP formulation. The remaining heats rely on empirical linear correlations, which are a slight departure from the fundamental treatment used for the blast heat.

4.3.5 Offgas Heat

The offgas heat computation is similar to the blast heat computation, employing proportionality constants that depends on the transition type. However, some thought must be given to the nature of the proportionality constants.

The offgas heat is generally given by

$$h^l_{\text{Offgas}} = \sum_{j \in \mathcal{S}_{\text{Offgas}}} \left[w_{\text{H}j}(T^{\text{Offgas}})\right] m^l_{j\text{Prod}}$$

For a linear program, the caveat is that the offgas temperature T^{Offgas} must somehow be constant, which may not be realistic. The MILP implementation is much better suited than a continuous linear program, as it permits piecewise linear computation,

$$h^l_{\text{Offgas}} = \begin{cases} \displaystyle\sum_{j \in \mathcal{S}_{\text{Offgas}}} w^{\text{Offgas}k}_{\text{H}j} m^l_{j\text{Prod}} & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\text{PSC}} \\ \\ 0 & \text{otherwise} \end{cases}$$

where $w^{\text{Offgas}k}_{\text{H}j} = w_{\text{H}j}(T^{\text{Offgas}k})$ are computed at nominal offgas temperatures $T^{\text{Offgas}k}$ which are associated to the transition type.

In industry, it is common to think of "the" offgas temperature $T^{\text{Offgas}k}$, which is typically 1200°C [66]. In actuality, the offgas temperature varies throughout the blowing action, as does the bath temperature. Thus $T^{\text{Offgas}k}$ may be regarded as a kind of empirical average over time, and similarly for $w^{\text{Offgas}k}_{\text{H}j}$. A more thorough treatment of the offgas heat is presented in Section 6.1, but it involves nonlinearities that are beyond the scope of the current MILP.

Thus, the current version of the MILP implements the following equations to estimate h^l_{Offgas},

$$h^l_{\text{Offgas}} \geq \sum_{j \in \mathcal{S}_{\text{Offgas}}} w^{\text{Offgas}k}_{\text{H}j} m^l_{j\text{Prod}} - \left(\overline{h}^l_{\text{Offgas}} - \underline{h}^l_{\text{Offgas}}\right)\left(1 - \beta^l_{\text{Type}k}\right) \tag{4.52}$$

$$h^l_{\text{Offgas}} \leq \sum_{j \in \mathcal{S}_{\text{Offgas}}} w^{\text{Offgas}k}_{\text{H}j} m^l_{j\text{Prod}} + \left(\overline{h}^l_{\text{Offgas}} - \underline{h}^l_{\text{Offgas}}\right)\left(1 - \beta^l_{\text{Type}k}\right) \tag{4.53}$$

for all $k \in \mathcal{T}_{\text{PSC}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$, in which the bounds can be taken as

$$\underline{h}_{\text{Offgas}} = \min\left(\min_{\substack{j \in \mathcal{S}_{\text{Offgas}} \\ k \in \mathcal{T}_{\text{PSC}}}} w^{\text{Offgas}k}_{\text{H}j} \overline{m}_{j\text{Prod}}, 0\right)$$

$$\overline{h}_{\text{Offgas}} = \max\left(\max_{\substack{j \in \mathcal{S}_{\text{Offgas}} \\ k \in \mathcal{T}_{\text{PSC}}}} w^{\text{Offgas}k}_{\text{H}j} \overline{m}_{j\text{Prod}}, 0\right)$$

In turn, these upper bounds depend on mass upper bounds $\overline{m}_{j\text{Prod}}$ for $j \in \mathcal{S}_{\text{Offgas}}$, which had not been considered previously. Firstly, the oxygen upper bound can be taken as

$$\overline{m}_{\text{O}_2,\text{Prod}} = (1 - e_{\text{O}})\overline{m}_{\text{O,Blast}}$$

which is related to Equation 4.46. The nitrogen upper bound is given by

$$\overline{m}_{\mathrm{N_2,Prod}} = \overline{m}_{\mathrm{N,Blast}}$$

with the understanding that the incoming nitrogen (blast) is entirely exhausted as outgoing nitrogen (offgas); there may be other feed streams that contain nitrogen, but this is assumed to be negligible compared to the blast nitrogen. The upper bound for SO_2 is given by

$$\overline{m}_{\mathrm{SO_2,Prod}} = \frac{\overline{m}_{\mathrm{O,Blast}}}{w_{\mathrm{O,SO_2}}}$$

which considers the case where none of the blast oxygen is consumed by a slag phase, as in the Copper-Blow reaction.

The case where l is undetermined is handled by the following equations,

$$h^l_{\mathrm{Offgas}} \geq \underline{h}_{\mathrm{Offgas}} \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \beta^l_{\mathrm{Type}k} \tag{4.54}$$

$$\leq \overline{h}_{\mathrm{Offgas}} \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \beta^l_{\mathrm{Type}k} \tag{4.55}$$

for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ$, so that $h^l_{\mathrm{Offgas}} = 0$ whenever l is undetermined.

Equations 4.52-55 are a practical structure, because they depend only on the offgas temperature, which is easily measured. But these equations are an approximation, as discussed in Section 6.1.

4.3.6 Discharge Heat

An exact treatment of discharge heats h^l_{DCh} would be nonlinear, as described in Section 6.1. Nonetheless, the MILP circumnavigates some of the nonlinear affects of h^l_{DCh}, by considering that $h^l_{\mathrm{Ret}} = 0$ when obj(l) is emptied (Equations 4.18-19).

The discharge heat is generally given by

$$h^l_{\mathrm{DCh}} = \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \left[w_{\mathrm{H}j}(T^{\mathrm{DCh}})\right] \left(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}}\right)$$

As in the previous subsection, the main complication is the variation of T^{DCh}. The idea of nominal temperatures $T^{\mathrm{DCh}k}$ can be adapted from the offgas temperature computation, hence $w^{\mathrm{DCh}k}_{\mathrm{H}j} = w_{\mathrm{H}j}(T^{\mathrm{DCh}k})$ would be constants associated with the transition type k.

It is undesirable, however, to fix the discharge temperatures $T^{\mathrm{DCh}k}$. In a copper converting aisle, for example, it may be acceptable for the blister copper to be discharged at any temperature between 1200°C and 1250°C, rather than artificially prescribing a fixed value. It is generally preferable for the MILP to admit bounds $\underline{T}^{\mathrm{DCh}k} \leq T^{\mathrm{DCh}l} \leq \overline{T}^{\mathrm{DCh}k}$, rather than artificially fixing $T^{\mathrm{DCh}l} = T^{\mathrm{DCh}k}$. The latter equality is a special case of $\underline{T}^{\mathrm{DCh}k} \leq T^{\mathrm{DCh}l} \leq \overline{T}^{\mathrm{DCh}k}$, in which $\underline{T}^{\mathrm{DCh}k} = \overline{T}^{\mathrm{DCh}k} = T^{\mathrm{DCh}k}$.

The current MILP formulation avoids the fixing of discharge temperatures as much as possible, while respecting linearity. The following computation has been successfully implemented,

$$h^l_{\mathrm{DCh}} = \begin{cases} \displaystyle\sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w^{\mathrm{DCh}k}_{\mathrm{H}j} \left(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}} \right) & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\mathrm{PSC,IDCh}} \\ \displaystyle\sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \left[w_{\mathrm{H}j}(T^{\mathrm{DCh}l}) \right] \left(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}} \right) & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\mathrm{PSC}} \setminus \mathcal{T}_{\mathrm{PSC,IDCh}} \\ 0 & \text{otherwise} \end{cases}$$

for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$. The formulation fixes discharge temperatures only for the transition types belonging to the subset $\mathcal{T}_{\mathrm{PSC,IDCh}} \subset \mathcal{T}_{\mathrm{PSC}}$, defined by

$$\mathcal{T}_{\mathrm{PSC,IDCh}} = \left\{ k \in \mathcal{T}_{\mathrm{PSC}} \setminus \mathcal{T}_{\mathrm{PSC,Empty}} \middle| \; \exists j \in \mathcal{F}_{\mathrm{DCh}} \text{ s.t. } \overline{v}^{jk} > 0 \right\}$$

This subset contains all transition types that include discharge actions, but which do not fully empty the converter, i.e. they do not end the converter cycle. These actions are considered "intermediate discharges" (IDCh), as opposed to the "final discharges" that do end the converter cycle.

Any final discharge action results in an empty converter so that $h^l_{\mathrm{Ret}} = 0$, regardless of what $T^{\mathrm{DCh}l}$ turns out to be (Equation 4.18-19). This allows the final discharge temperature for blister copper, or for iron-free nickel matte, to be determined as part of the optimization. Moreover, $T^{\mathrm{DCh}l}$ may be subject to lower and upper bounds, as described in Subsection 4.5.4.

Unfortunately, the MILP formulation is unable to provide the same freedom for the intermediate discharge actions, forcing prescribed temperatures $T^{\mathrm{DCh}l} = T^{\mathrm{DCh}k}$ whenever l is of a type $k \in \mathcal{T}_{\mathrm{PSC,IDCh}}$. In practice, this means that intermediate skimming temperatures must be predetermined, instead of resulting from the optimization. This drawback will be revisited in Section 5.3, and again in Section 6.1. For the skimming of iron-bearing slag,

$T^{\mathrm{DCh}k} = 1230°\mathrm{C}$ is a typical bath temperature [7, 66].

When there is no discharge action, $(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}}) = 0$ for all $j \in \mathcal{S}_{\mathrm{NGProd}}$, so that h^l_{DCh} is automatically zero. This assertion extends to undetermined transitions, which forbid any discharging according to Equation 4.23. In either of these cases, $h^l_{\mathrm{DCh}} = 0$.

The discharge heat computation is incorporated into the MILP for intermediate discharges,

$$h^l_{\mathrm{DCh}} \geq \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w^{\mathrm{DCh}k}_{\mathrm{H}j} \left(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}}\right) - \left(\overline{h}^l_{\mathrm{DCh}} - \underline{h}^l_{\mathrm{DCh}}\right)\left(1 - \beta^l_{\mathrm{Type}k}\right) \tag{4.56}$$

$$\leq \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w^{\mathrm{DCh}k}_{\mathrm{H}j} \left(m^l_{j\mathrm{Prod}} - m^l_{j\mathrm{RetProd}}\right) + \left(\overline{h}^l_{\mathrm{DCh}} - \underline{h}^l_{\mathrm{DCh}}\right)\left(1 - \beta^l_{\mathrm{Type}k}\right) \tag{4.57}$$

for all $k \in \mathcal{T}_{\mathrm{PSC,IDCh}}$ and $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ$, in which the bounds can be taken as

$$\underline{h}_{\mathrm{DCh}} = \min\left(\min_{k \in \mathcal{T}_{\mathrm{PSC}}} \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \underline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} \overline{m}^l_{j\mathrm{Prod}}, 0\right)$$

$$\overline{h}_{\mathrm{DCh}} = \max\left(\max_{k \in \mathcal{T}_{\mathrm{PSC}}} \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \overline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} \overline{m}^l_{j\mathrm{Prod}}, 0\right)$$

in which $\overline{m}^l_{j\mathrm{Prod}}$ is computed as for Equation 4.42. The specific heat bounds are taken as $\underline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} = w_{\mathrm{H}j}(\underline{T}^{\mathrm{DCh}k})$ and $\overline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} = w_{\mathrm{H}j}(\overline{T}^{\mathrm{DCh}k})$, where $\underline{T}^{\mathrm{DCh}k}$ and $\overline{T}^{\mathrm{DCh}k}$ are defining parameters for all $k \in \mathcal{T}_{\mathrm{PSC}}$. Whenever $k \in \mathcal{T}_{\mathrm{PSC,IDCh}}$, it is imperative that $\underline{T}^{\mathrm{DCh}k} = \overline{T}^{\mathrm{DCh}k} = T^{\mathrm{DCh}k}$, so that the intermediate discharge temperatures are indeed fixed.

The case that l does not include any discharging, or is undetermined, is handled by the following equations,

$$h^l_{\mathrm{DCh}} \geq \underline{h}_{\mathrm{DCh}} \sum_{k \in \mathcal{T}_{\mathrm{PSC,Empty}} \cup \mathcal{T}_{\mathrm{PSC,IDCh}}} \beta^l_{\mathrm{Type}k} \tag{4.58}$$

$$\leq \overline{h}_{\mathrm{DCh}} \sum_{k \in \mathcal{T}_{\mathrm{PSC,Empty}} \cup \mathcal{T}_{\mathrm{PSC,IDCh}}} \beta^l_{\mathrm{Type}k} \tag{4.59}$$

for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ$.

Equations 4.56-59 do not impose any restrictions regarding final discharges. However this case is covered by the Equations 4.18-19, in conjunction with the forward heat computation (Subsection 4.4.3) and the discharge temperature bounds (Subsection 4.4.5).

4.3.7 Environmental Heat Losses

The heat losses to the environment $h^l_{\text{Env}i}$ are treated in a similar manner as the blast heat (Subsection 4.3.3). There is a linear relationship between the heat loss and the segment duration d^l_i, although the slope and intercept depends on the transition type.

The environmental heat losses $h^l_{\text{Env}\circ}$ which occur between transitions $l-$ and l, depend on the mechanistic state left by $l-$. Therefore,

$$h^l_{\text{Env}\circ} = \begin{cases} h^k_{\text{Env}\circ} + \dot{h}^k_{\text{Env}\circ} d^l_\circ & \text{if } l- \text{ is of a type } k \in \mathcal{T}_{\text{PSC}} \\ 0 & \text{otherwise} \end{cases}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. The other segments depend on the current transition,

$$h^l_{\text{Env}i} = \begin{cases} h^k_{\text{Env}i} + \dot{h}^k_{\text{Env}i} d^l_i & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\text{PSC}} \\ 0 & \text{otherwise} \end{cases}$$

for all $i \in \{1, 2, \ldots, 7\}$, and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. For the moment, $h^k_{\text{Env}i}$ and $\dot{h}^k_{\text{Env}i}$ can be regarded as empirically observed parameters, but they will be revisited in Section 6.1. If a transition type $k \in \mathcal{T}_{\text{PSC}}$ is such that $\overline{d}^k_i = 0$, then $\dot{h}^k_{\text{Env}i}$ can be arbitrarily set to zero.

The environmental heat losses are taken to be nonnegative, $h^l_{\text{Env}i} \in \mathbb{R}^+_\circ$. It is a rather modest assumption that heat should flow from the hot converters into the colder environment, regardless of the value of d^l_i. To be consistent with this assumption, the parameters must be nonnegative, $h^k_{\text{Env}i} \geq 0$ and $\dot{h}^k_{\text{Env}i} \geq 0$. Also, $h^k_{\text{Env}7} = \dot{h}^k_{\text{Env}7} = 0$ for all $k \in \mathcal{T}_{\text{PSC,Empty}}$, because no heat can be drawn from the bath after the converter has already been emptied of its bath.

The inter-transition heat loss is implemented through the following inequalities,

$$\begin{aligned} h^l_{\text{Env}\circ} &\geq h^k_{\text{Env}\circ}\beta^{l-}_{\text{Type}k} + \dot{h}^k_{\text{Env}\circ} d^l_\circ - \overline{h}_{\text{Env}\circ}\left(1 - \beta^{l-}_{\text{Type}k}\right) && (4.60) \\ &\leq h^k_{\text{Env}\circ}\beta^{l-}_{\text{Type}k} + \dot{h}^k_{\text{Env}\circ} d^l_\circ + \overline{h}_{\text{Env}\circ}\left(1 - \beta^{l-}_{\text{Type}k}\right) && (4.61) \end{aligned}$$

for all $k \in \mathcal{T}_{\text{PSC}}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Equations 4.60-61 are slack when $\beta^{l-}_{\text{Type}k} = 0$, otherwise they force the desired linear relationship $h^l_{\text{Env}\circ} = h^k_{\text{Env}\circ} + \dot{h}^k_{\text{Env}\circ} d^l_\circ$. The environmental losses that occur during the transitions are implemented through

$$\begin{aligned} h^l_{\text{Env}i} &\geq h^k_{\text{Env}i}\beta^{l}_{\text{Type}k} + \dot{h}^k_{\text{Env}i} d^l_i - \overline{h}_{\text{Env}i}\left(1 - \beta^{l}_{\text{Type}k}\right) && (4.62) \\ &\leq h^k_{\text{Env}i}\beta^{l}_{\text{Type}k} + \dot{h}^k_{\text{Env}i} d^l_i + \overline{h}_{\text{Env}i}\left(1 - \beta^{l}_{\text{Type}k}\right) && (4.63) \end{aligned}$$

for $i \in \{1,2,\ldots,7\}$, $k \in \mathcal{T}_{\text{PSC}}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. Equations 4.62-63 are slack when $\beta^l_{\text{Type}k} = 0$, otherwise they force the desired linear relationship $h^l_{\text{Env}i} = h^k_{\text{Env}i} + \dot{h}^k_{\text{Env}i} d^l_i$. The upper bounds are given by

$$\overline{h}_{\text{Env}i} = \max_{k \in \mathcal{T}_{\text{PSC}}} \left(h^k_{\text{Env}i} + \dot{h}^k_{\text{Env}i} \overline{d}^k_i \right)$$

for all $i \in \{0,1,\ldots,7\}$.

The environmental heat loss should be set to zero for undetermined transitions,

$$h^l_{\text{Env}i} \leq \overline{h}_{\text{Env}i} \sum_{k \in \mathcal{T}_{\text{PSC}}} \beta^l_{\text{Type}k} \tag{4.64}$$

for $i \in \{0,1,\ldots,7\}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$.

The development of Section 4.3 is particular to the MILP implementation of the PSC Problem. In future work, other models can be developed outside of the strict MILP format, in which part or all of Equations 4.39-64 might be replaced. Meanwhile, the current set of intermediate computations successfully support the state-machine representation of PS converting, within an optimization framework.

4.4 Forward State Computation

4.4.1 Retained Feed Masses

The current formulation assumes that the feed streams are either completely reacted, or completely retained, depending on whether or not a blowing action is performed. Any feed streams that are present at the onset of a blowing action are immediately mixed into the product streams, and presumed to be completed reacted.

This complete-reaction condition can be stated as

$$m^l_{\text{Ret}k} = \begin{cases} 0 & \text{if transition } l \text{ is of a type that includes a blow} \\ m^l_k & \text{otherwise} \end{cases}$$

and is implemented through the following inequalities,

$$m^l_{\text{Ret}k} \geq m^l_k - \overline{m}_k \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \bar{d}^{k'}_{\text{Blow}} > 0}} \beta^l_{\text{Type}k'} \tag{4.65}$$

$$\leq m^l_k + \overline{m}_k \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \bar{d}^{k'}_{\text{Blow}} > 0}} \beta^l_{\text{Type}k'} \tag{4.66}$$

$$\leq \overline{m}_k \left(1 - \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \bar{d}^{k'}_{\text{Blow}} > 0}} \beta^l_{\text{Type}k'} \right) \tag{4.67}$$

for all $k \in \mathcal{Z}_{\text{NGFeed}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$, where the upper bounds are given by

$$\overline{m}_k = \rho_k \max_{j \in \{1,\ldots,n_{\text{PSC}}\}} \overline{v}^{\text{PSC}j}$$

From the complete-reaction condition, it is apparent that $m^l_{\text{Ret}k} \leq m^l_k$.

Molten and granular feeds are rapidly incorporated into the product streams, so that the complete-reaction condition is especially appropriate for these feeds. On the other hand, a detailed representation of copper scrap, recycled metal, etc. could involve the rate of reaction $\dot{\xi}_k$, which is related to the coefficients ξ_k that had been introduced in Subsection 3.4.2. Such a formulation could be supported by the MILP structure, if the reaction rates depend only on the transition types. Nonetheless, the complete-reaction condition poses a simple forward computation that is adequate for the current formulation.

4.4.2 Retained Product Species Masses

To compute the retained product masses, the current formulation depends on a complete-discharge condition. Each product stream is are either completely discharged, or completely retained. This complete-discharge condition is similar to the complete-reaction condition.

The complete-discharge condition can be stated succinctly,

$$m^l_{j\text{RetProd}} = \begin{cases} 0 & \text{if transition } l \text{ is of a type that includes a discharge of stream } k \\ m^l_{j\text{Prod}} & \text{otherwise} \end{cases}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$ and $j \in \mathcal{S}_k$. It is incorporated into the MILP through the following three

inequalities,

$$m^l_{j\text{RetProd}} \geq m^l_{j\text{Prod}} - \overline{m}_{j\text{Prod}} \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \exists j' \in \mathcal{F}_k \text{s.t.} \overline{v}^{j'k'} > 0}} \beta^l_{\text{Type}k'} \tag{4.68}$$

$$\leq m^l_{j\text{Prod}} + \overline{m}_{j\text{Prod}} \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \exists j' \in \mathcal{F}_k \text{s.t.} \overline{v}^{j'k'} > 0}} \beta^l_{\text{Type}k'} \tag{4.69}$$

$$\leq \overline{m}_{j\text{Prod}} \left(1 - \sum_{\substack{k' \in \mathcal{T}_{\text{PSC}} \\ \exists j' \in \mathcal{F}_k \text{s.t.} \overline{v}^{j'k'} > 0}} \beta^l_{\text{Type}k'} \right) \tag{4.70}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$, $j \in \mathcal{S}_k$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$, in which $\overline{m}_{j\text{Prod}}$ are computed as for Equation 4.42. From the complete-discharge condition, it is apparent that $m^l_{j\text{RetProd}} \leq m^l_{j\text{Prod}}$.

The discharged product masses are related to the discharge flows, through the discharge volume balance,

$$\sum_{j \in \mathcal{S}_k} \frac{m^l_{j\text{Prod}} - m^l_{j\text{RetProd}}}{\rho_j} = \sum_{j \in \mathcal{F}_k} v^{jl} \tag{4.71}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. In conjunction with Equations 4.30-31, this ensures that enough carrying devices are provided in case stream k is discharged.

The complete-discharge condition is appropriate for the MILP formulation, since smelters do not usually plan partial discharges. For instance, a decision to skim the slag implies the entire removal of all of the slag. Similarly, the final discharge implies the removal of all of the final product, be it blister copper or converter matte. Nonetheless, the formulation would provide more realism if the complete-discharge condition could be relaxed, as discussed in Section 6.1.

4.4.3 Retained Heat

The heat retained within the bath is distributed throughout the retained mass, and is what determines the bath temperature. The forward heat computation is therefore essential in establishing the temperature constraints of Subsection 4.5.4.

The retained heat is computed as,

$$h^l_{\text{Ret}} = h^{l-}_{\text{Ret}} + h^l_{\text{Ch}} + h^l_{\text{NGBlow}} + h^l_{\text{Blast}} - h^l_{\text{Offgas}} - h^l_{\text{DCh}} - \sum_{i=0}^{7} h^l_{\text{Env}i} \tag{4.72}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. This includes the heat retained from the previous transition, plus the heat introduced by the feed from the current transition ($h^l_{\text{Ch}} + h^l_{\text{NGBlow}} + h^l_{\text{Blast}}$), minus the heat which is removed as product streams ($h^l_{\text{Offgas}} + h^l_{\text{DCh}}$), and minus the environmental losses.

The discharge heat h^l_{DCh} has been deliberately left variable in case the converter is empty or is being emptied (Equations 4.58-59). When l is of a type belonging to $\mathcal{T}_{\text{PSC,Empty}}$, Equations 4.18-19 force $h^l_{\text{Ret}} = 0$, so that

$$h^l_{\text{DCh}} = h^{l-}_{\text{Ret}} + h^l_{\text{Ch}} + h^l_{\text{NGBlow}} + h^l_{\text{Blast}} - h^l_{\text{Offgas}} - \sum_{i=0}^{6} h^l_{\text{Env}i}$$

As discussed in Subsection 4.3.7, the proper interpretation of $\mathcal{T}_{\text{PSC,Empty}}$ would require that $h^l_{\text{Env7}} = 0$. Thus a final discharge causes h^l_{DCh} to take away any heat that had been in the vessel. (The heat of content of the refractory bricks is negligible compared to the heat content of the bath).

Equation 4.72 is itself correct, regardless of how the individual terms are computed, and regardless of whether the complete-reaction condition (Subsection 4.4.1) or the complete-discharge condition (Subsection 4.4.2) are held. This robustness can be helpful in future work, in case there are changes in the intermediate computations, or the forward mass computations.

4.5 Feasible Converter Transitions

4.5.1 Direct Transition Constraints in General Linear Form

In order for a transition l to be feasible, the proposed transition variables (l) must be self-consistent, and must agree with the preceding state ($l-$). Herein, linear inequalities of the preceding state variables and the current transition variables can be directly implemented into the MILP.

For instance, Equation 4.20 considers only the transition-type determinants to verify the mechanistic preparedness of the converter. Equations 4.21-34 are constraints which do not consider the preceding state variables, thus focusing on the self-consistency of the current transition variables.

$\mathcal{L}^{\text{DTrans}}_{\text{PSC}}$ is the set of transition feasibility clauses that are implemented directly as linear

inequalities under the following linear form.

$$\begin{aligned} &\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} a^i_{m\text{Ret-},k} m^{l-}_{\text{Ret}k} \quad + \quad \sum_{j\in\mathcal{S}_{\text{NGProd}}} a^i_{m,j\text{RetProd-}} m^{l-}_{j\text{RetProd}} \quad + \quad a^i_{h\text{Ret-}} h^{l-}_{\text{Ret}} \\ + \quad &\sum_{k\in\mathcal{T}_{\text{PSC}}} a^i_{\beta\text{-},k} \beta^{l-}_{\text{Type}k} \quad + \quad \sum_{i'=0}^{7} a^i_{d,i'} d^l_{i'} \quad + \quad \sum_{j\in\mathcal{F}_{\text{NG}}} a^i_{v,j} v^{jl} \\ + \quad &\sum_{j\in\mathcal{F}^{\text{MSM}}} a^i_{u,j} u^{jl} \quad + \quad \sum_{k\in\mathcal{T}_{\text{PSC}}} a^i_{\beta,k} \beta^l_{\text{Type}k} \quad \leq \quad b^i \end{aligned} \tag{4.73}$$

for all $i \in \mathcal{L}^{\text{DTrans}}_{\text{PSC}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Each clause i is characterized by the coefficients, $a^i_{m\text{Ret-},k}$, $a^i_{m,j\text{RetProd-}}$, $a^i_{h\text{Ret-}}$, $a^i_{\beta\text{-},k}$, $a^i_{d,i'}$, $a^i_{v,j}$, $a^i_{u,j}$, and $a^i_{\beta,k}$, and by the righthand constant b^i.

Equation 4.20 can be expressed in general form, by setting $a^i_{\beta,k} = 1$ for some $k \in \mathcal{T}_{\text{PSC}}$, $a^i_{\beta\text{-},k'} = -1$ for all $k' \in \mathcal{T}^-_{\text{PSC}k}$, and all remaining parameters equal to zero. The same approach can be taken for Equations 4.17-19 and 4.21-29, as well as Equations 4.31 and 4.34, always bringing the variables to the lefthand side of "$\leq$". As described in Appendix B.1, equalities are the intersection between two inequalities. For example, Equation 4.30 can be reexpressed

$$\begin{aligned} v^{jl} - v^j_u u^{jl} &\leq 0 \\ -v^{jl} + v^j_u u^{jl} &\leq 0 \end{aligned}$$

Equations 4.32-33 can be treated similarly. Equations 4.17-34 can all be expressed as members of $\mathcal{L}^{\text{DTrans}}_{\text{PSC}}$. They have nonetheless been implemented separately, for instructive purposes.

Blending conditions are a *bona fide* application of Equation 4.73 that has not yet been considered. For example, there may be a nongaseous feed mechanism $j \in \mathcal{F}_{\text{NGFeed}}$ that must always be applied in conjunction with another mechanism $j' \in \mathcal{F}_{\text{NGFeed}}$, such that

$$v^{jl} \leq (0.25) v^{j'l}$$

In this case, the volume delivered by mechanism j cannot exceed 25% of the volume delivered by mechanism j'. More elaborate blending conditions are possible.

Equation 4.73 is sufficiently general that it can support any linear constraint that relates the preceding converter state $(m^{l-}_{\text{Ret}k}, m^{l-}_{j\text{RetProd}}, h^{l-}_{\text{Ret}}, \beta^{l-}_{\text{Type}k})$ to the current transition $(d^l_i, v^{jl}, u^{jl}, \beta^l_{\text{Type}k})$. Of course, the exact nature of Equation 4.73 depends on the members of $\mathcal{L}^{\text{DTrans}}_{\text{PSC}}$, which ultimately depends on the particular context.

However, there are constraints which must be respected by the evolving converter, but which cannot be formulated directly as linear relationships between the state and transition variables, and thus do not conform to Equation 4.73. These are indirect transition constraints, and are described in the remainder of Section 4.5. Such constraints depend on essential intermediate variables that have been implemented in accordance with Section 4.3.

4.5.2 Bath Composition Constraints

For a converter to undergo a certain transition, the bath must be of an appropriate composition. For example, a Copper-Blow is not permitted unless all of the iron has been removed, as discussed in Subsection 1.3.2.

The MILP formulation places upper and lower bounds on the mass-fractions that describe the overall composition of the bath, and that describe the composition of the individual product streams. As depicted in Figure 4.2, the compositions are monitored at three different times within the generic converter transition,

- Prior to charging, which coincides with the beginning of the transition
- Following blowing, which coincides with the end of the bulk chemical reactions
- Following discharging, which coincides with the end of the transition

The formulation does not distinguish between the composition of an individual product stream before and after discharging, because the streams are presumed to be homogeneous, due to the favourable mixing characteristics of PS converting [7].

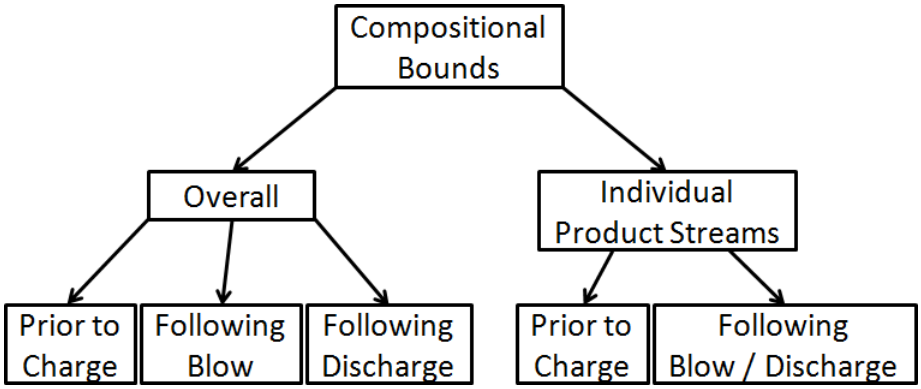


Figure 4.2: Bounds are placed on the overall bath composition and individual stream composition, placing restrictions prior to charging, following blowing and following discharging

The overall mass-fraction of element i within the bath is denoted $w_i^{\circ l}$, $w_i^{\mathrm{Blow}l}$ or $w_i^{\mathrm{DCh}l}$, depending on whether the measurement is prior to the charging, following the blowing or following the discharging of transition $l \in \mathcal{A}_{\mathrm{PSC}} \setminus A_\circ$, for all $i \in \mathcal{E}$. These are related to the previously defined masses,

$$
\begin{aligned}
w_i^{\circ l} &= \frac{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{RetProd}}^{l-}}{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{RetProd}}^{l-}} \\
w_i^{\mathrm{Blow}l} &= \frac{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{Prod}}^{l}}{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k}^{l} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{Prod}}^{l}} \\
w_i^{\mathrm{DCh}l} &= \frac{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{RetProd}}^{l}}{\displaystyle\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k}^{l} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{RetProd}}^{l}}
\end{aligned}
$$

The three expressions do not rely on the complete-reaction condition that was described in Subsection 4.4.1, but they do assume that feed streams are reacted only during the blowing actions.

The MILP implementation does not make assumptions about the speciation of feed streams, which may be retained in the bath. Therefore, it is generally not possible to implement bounds on the species compositions of the bath. However, the overall mass-fraction of stream k within the bath is denoted $w_{k\circ}^{l}$, $w_{k\mathrm{Blow}}^{l}$ or $w_{k\mathrm{DCh}}^{l}$, adopting the same superscripts as for the elemental fractions. These stream-based fractions are given by

$$
w_k^{\circ l} = \begin{cases}
\dfrac{m_{\mathrm{Ret}k}^{l-}}{\displaystyle\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k'}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{RetProd}}^{l-}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGFeed}} \\
\dfrac{\displaystyle\sum_{j \in \mathcal{S}_k} m_{j\mathrm{RetProd}}^{l-}}{\displaystyle\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k'}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{RetProd}}^{l-}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGProd}}
\end{cases}
$$

$$w_k^{\mathrm{Blow}l} = \begin{cases} \dfrac{m^l_{\mathrm{Ret}k}}{\sum\limits_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum\limits_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{Prod}}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGFeed}} \\ \dfrac{\sum\limits_{j \in \mathcal{S}_k} m^l_{j\mathrm{Prod}}}{\sum\limits_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum\limits_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{Prod}}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGProd}} \end{cases}$$

$$w_k^{\mathrm{DCh}l} = \begin{cases} \dfrac{m^l_{\mathrm{Ret}k}}{\sum\limits_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum\limits_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{RetProd}}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGFeed}} \\ \dfrac{\sum\limits_{j \in \mathcal{S}_k} m^l_{j\mathrm{RetProd}}}{\sum\limits_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum\limits_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{RetProd}}} & \text{if } k \in \mathcal{Z}_{\mathrm{NGProd}} \end{cases}$$

Once again, these expressions assume that the feed streams are reacted only during the blowing actions.

The elemental composition of individual product streams is developed in a similar way as for the overall bath,

$$w_{ik}^{\circ l} = \frac{\sum\limits_{j \in \mathcal{S}_k} w_{ij} m^{l-}_{j\mathrm{RetProd}}}{\sum\limits_{j \in \mathcal{S}_k} m^{l-}_{j\mathrm{RetProd}}}$$

$$w_{ik}^{\mathrm{Blow}l} = \frac{\sum\limits_{j \in \mathcal{S}_k} w_{ij} m^{l}_{j\mathrm{Prod}}}{\sum\limits_{j \in \mathcal{S}_k} m^{l}_{j\mathrm{Prod}}}$$

for all $i \in \mathcal{E}$, $k \in \mathcal{Z}_{\mathrm{NGProd}}$, and $l \in \mathcal{A}_{\mathrm{PSC}} \setminus A_\circ$.

The species mass-fractions of product streams are given by

$$w_{jk}^{\circ l} = \frac{m_{j\mathrm{RetProd}}^{l-}}{\sum_{j' \in \mathcal{S}_k} m_{j'\mathrm{RetProd}}^{l-}}$$

$$w_{jk}^{\mathrm{Blow}l} = \frac{m_{j\mathrm{Prod}}^{l}}{\sum_{j' \in \mathcal{S}_k} m_{j'\mathrm{Prod}}^{l}}$$

for all $k \in \mathcal{Z}_{\mathrm{NGProd}}$, all $j \in \mathcal{S}_k$. Naturally, $w_{jk}^{\circ l} = w_{jk}^{\mathrm{Blow}l} = 0$ when $j \in \mathcal{S}_{\mathrm{Prod}} \setminus \mathcal{S}_k$ is a species which does not report to stream k. Due to the homogeneity of product streams, it is not necessary to consider $w_{ik}^{\mathrm{DCh}l}$ and $w_{jk}^{\mathrm{DCh}l}$, because $w_{ik}^{\mathrm{DCh}l} = w_{ik}^{\mathrm{Blow}l}$ and $w_{jk}^{\mathrm{DCh}l} = w_{jk}^{\mathrm{Blow}l}$.

In summary there are ten kinds of mass-fractions ($w_i^{\circ l}$, $w_i^{\mathrm{Blow}l}$, $w_i^{\mathrm{DCh}l}$, $w_k^{\circ l}$, $w_k^{\mathrm{Blow}l}$, $w_k^{\mathrm{DCh}l}$, $w_{ik}^{\circ l}$, $w_{ik}^{\mathrm{Blow}l}$, $w_{jk}^{\circ l}$, $w_{jk}^{\mathrm{Blow}l}$) that are subject to lower and upper bounds. For transition l to be of type $k' \in \mathcal{T}_{\mathrm{PSC}}$,

$$\underline{w}_i^{\circ k'} \leq w_i^{\circ l} \leq \overline{w}_i^{\circ k'}$$

$$\underline{w}_i^{\mathrm{Blow}k'} \leq w_i^{\mathrm{Blow}l} \leq \overline{w}_i^{\mathrm{Blow}k'}$$

etc., where the bounds ($\underline{w}_i^{\circ k'}$, $\overline{w}_i^{\circ k'}$, $\underline{w}_i^{\mathrm{Blow}k'}$, $\overline{w}_i^{\mathrm{Blow}k'}$, …, $\underline{w}_{jk}^{\circ k'}$, $\overline{w}_{jk}^{\circ k'}$, $\underline{w}_{jk}^{\mathrm{Blow}k'}$, $\overline{w}_{jk}^{\mathrm{Blow}k'}$) are parameters that describe transition type k'.

Incidently, $\overline{w}_{\mathrm{Fe}}^{\circ k} = 0$ if k is a Copper-Blow transition type; this forbids the application of Copper-Blow transitions unless all of the iron has been removed. Depending on how the other transition types are parametrized, it may worthwhile to eliminate the slag stream as well, by setting $\overline{w}_{\mathrm{Slag}}^{\circ k} = 0$.

To implement the compositional bounds within the MILP, it is necessary to multiply through by the denominator. For example, $\underline{w}_i^{\circ k} \leq w_i^{\circ l} \leq \overline{w}_i^{\circ k}$ is expressed in terms of masses,

$$\underline{w}_i^{\circ k} \leq \frac{\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{RetProd}}^{l-}}{\sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} m_{\mathrm{Ret}k}^{l-} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m_{j\mathrm{RetProd}}^{l-}} \leq \overline{w}_i^{\circ k}$$

Multiplying through by the denominator gives two linear inequalities,

$$\begin{aligned}\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} w_{ik}m^{l-}_{\text{Ret}k} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} w_{ij}m^{l-}_{j\text{RetProd}} &\geq \underline{w}_i^{\circ k}\left(\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}}\right)\\ &\leq \overline{w}_i^{\circ k}\left(\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}}\right)\end{aligned}$$

Additional terms must be included, however, to control the slackness of these equations,

$$\begin{aligned}\sum_{k'\in\mathcal{Z}_{\text{NGFeed}}} w_{ik}m^{l-}_{\text{Ret}k'} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} w_{ij}m^{l-}_{j\text{RetProd}} &\geq \underline{w}_i^{\circ k}\left(\sum_{k'\in\mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k'} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}}\right)\\ &\quad -\overline{m}_i\left(1-\beta^l_{\text{Type}k}\right) \qquad (4.74)\\ &\leq \overline{w}_i^{\circ k}\left(\sum_{k'\in\mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k'} + \sum_{j\in\mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}}\right)\\ &\quad +\overline{m}_i\left(1-\beta^l_{\text{Type}k}\right) \qquad (4.75)\end{aligned}$$

for all $i \in \mathcal{E}$, $k \in \mathcal{T}_{\text{PSC}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. The factor $\overline{m}_i$ is given by

$$\overline{m}_i = \max\left(\max_{k\in\mathcal{Z}_{\text{NGFeed}}} w_{ik}\overline{m}_k, \max_{j\in\mathcal{S}_{\text{NGProd}}} w_{ij}\overline{m}_{j\text{Prod}}\right) + \overline{m}_{i\text{Blast}}$$

in which $\overline{m}_k$, $\overline{m}_{j\text{Prod}}$ and $\overline{m}_{i\text{Blast}}$ are computed as for Equations 4.65-67, Equation 4.42, and Equations 4.39-40, respectively. When $\beta^l_{\text{Type}k} = 1$, Equations 4.74-75 lead to the desired result that $\underline{w}_i^{\circ k} \leq w_i^{\circ l} \leq \overline{w}_i^{\circ k}$. Otherwise, $\overline{m}_{\text{Ret}i}$ is such that Equations 4.74-75 are slack when $\beta^l_{\text{Type}k} = 0$.

Equation 4.74 is redundant when $\underline{w}_i^{\circ k} = 0$, because the nonnegativity of the mass variables already ensures that $w_i^{\circ l} \geq 0$. Similarly, Equation 4.75 is redundant for $\overline{w}_i^{\circ k} \geq \max\left(\max_{k'\in\mathcal{S}_{\text{NGFeed}}} w_{ik'}, \max_{j\in\mathcal{S}_{\text{NGProd}}} w_{ij}\right)$ is automatically implied by the other constraints. These limits thus provide appropriate defaults as given in Table 4.2.

Table 4.2: Default values for mass-fraction bounds

Bounds	Default Values
$\left(\quad \underline{w}_i^{\circ k} \quad , \quad \overline{w}_i^{\circ k} \quad \right)$	$\left(\quad 0 \quad , \quad \max\left(\max_{k' \in \mathcal{S}_{\mathrm{NGFeed}}} w_{ik'}, \max_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} \right) \quad \right)$
$\left(\quad \underline{w}_i^{\mathrm{Blow}k} \quad , \quad \overline{w}_i^{\mathrm{Blow}k} \quad \right)$	$\left(\quad 0 \quad , \quad \max\left(\max_{k' \in \mathcal{S}_{\mathrm{NGFeed}}} w_{ik'}, \max_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} \right) \quad \right)$
$\left(\quad \underline{w}_i^{\mathrm{DCh}k} \quad , \quad \overline{w}_i^{\mathrm{DCh}k} \quad \right)$	$\left(\quad 0 \quad , \quad \max\left(\max_{k' \in \mathcal{S}_{\mathrm{NGFeed}}} w_{ik'}, \max_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} \right) \quad \right)$
$\left(\quad \underline{w}_k^{\circ k'} \quad , \quad \overline{w}_k^{\circ k'} \quad \right)$	$\left(\quad 0 \quad , \quad 1 \quad \right)$
$\left(\quad \underline{w}_k^{\mathrm{Blow}k'} \quad , \quad \overline{w}_k^{\mathrm{Blow}k'} \quad \right)$	$\left(\quad 0 \quad , \quad 1 \quad \right)$
$\left(\quad \underline{w}_k^{\mathrm{DCh}k'} \quad , \quad \overline{w}_k^{\mathrm{DCh}k'} \quad \right)$	$\left(\quad 0 \quad , \quad 1 \quad \right)$
$\left(\quad \underline{w}_{ik}^{\circ k'} \quad , \quad \overline{w}_{ik}^{\circ k'} \quad \right)$	$\left(\quad \min_{j \in \mathcal{S}_k} w_{ij} \quad , \quad \max_{j \in \mathcal{S}_k} w_{ij} \quad \right)$
$\left(\quad \underline{w}_{ik}^{\mathrm{Blow}k'} \quad , \quad \overline{w}_{ik}^{\mathrm{Blow}k'} \quad \right)$	$\left(\quad \min_{j \in \mathcal{S}_k} w_{ij} \quad , \quad \max_{j \in \mathcal{S}_k} w_{ij} \quad \right)$
$\left(\quad \underline{w}_{jk}^{\circ k'} \quad , \quad \overline{w}_{jk}^{\circ k'} \quad \right)$	$\left(\quad 0 \quad , \quad 1 \quad \right)$
$\left(\quad \underline{w}_{jk}^{\mathrm{Blow}k'} \quad , \quad \overline{w}_{jk}^{\mathrm{Blow}k'} \quad \right)$	$\left(\quad 0 \quad , \quad 1 \quad \right)$

The remaining bounds are implemented in a similar manner, again by multiplying through by the denominator, and by including binary terms to control the slackness. For all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$,

$$\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} w_{ik'} m^l_{\mathrm{Ret}k'} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m^l_{j\mathrm{Prod}} \geq \underline{w}_i^{\mathrm{Blow}k} \left(\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{Prod}} \right) - \overline{m}_i \left(1 - \beta^l_{\mathrm{Type}k} \right) \tag{4.76}$$

$$\leq \overline{w}_i^{\mathrm{Blow}k} \left(\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} m^l_{\mathrm{Ret}k'} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} m^l_{j\mathrm{Prod}} \right) + \overline{m}_i \left(1 - \beta^l_{\mathrm{Type}k} \right) \tag{4.77}$$

for all $i \in \mathcal{E}$ and $k \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} w_{ik'} m^l_{\text{Ret}k'} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} w_{ij} m^l_{j\text{RetProd}} \geq \underline{w}_i^{\text{DCh}k} \left(\sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k'} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) - \overline{m}_i \left(1 - \beta^l_{\text{Type}k}\right) \tag{4.78}$$

$$\leq \overline{w}_i^{\text{DCh}k} \left(\sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k'} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) + \overline{m}_i \left(1 - \beta^l_{\text{Type}k}\right) \tag{4.79}$$

for all $i \in \mathcal{E}$ and $k \in \mathcal{T}_{\text{PSC}}$,

$$m^{l-}_{\text{Ret}k} \geq \underline{w}_k^{\circ k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.80}$$

$$\leq \overline{w}_k^{\circ k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.81}$$

for all $k \in \mathcal{Z}_{\text{NGFeed}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{j \in \mathcal{S}_k} m^{l-}_{j\text{RetProd}} \geq \underline{w}_k^{\circ k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.82}$$

$$\leq \overline{w}_k^{\circ k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^{l-}_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^{l-}_{j\text{RetProd}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.83}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$m^l_{\text{Ret}k} \geq \underline{w}_k^{\text{Blow}k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{Prod}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.84}$$

$$\leq \overline{w}_k^{\text{Blow}k'} \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{Prod}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \tag{4.85}$$

for all $k \in \mathcal{Z}_{\text{NGFeed}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{j \in \mathcal{S}_k} m^l_{j\text{Prod}} \geq \underline{w}^{\text{Blow}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{Prod}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.86)$$

$$\leq \overline{w}^{\text{Blow}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{Prod}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.87)$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$m^l_{\text{Ret}k} \geq \underline{w}^{\text{DCh}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.88)$$

$$\leq \overline{w}^{\text{DCh}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.89)$$

for all $k \in \mathcal{Z}_{\text{NGFeed}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{j \in \mathcal{S}_k} m^l_{j\text{RetProd}} \geq \underline{w}^{\text{DCh}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) - \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.90)$$

$$\leq \overline{w}^{\text{DCh}k'}_k \left(\sum_{k'' \in \mathcal{Z}_{\text{NGFeed}}} m^l_{\text{Ret}k''} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} m^l_{j\text{RetProd}} \right) + \overline{m}_k \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.91)$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$ and $k' \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{j \in \mathcal{S}_k} w_{ij} m^{l-}_{j\text{RetProd}} \geq \underline{w}^{\circ k'}_{ik} \sum_{j \in \mathcal{S}_k} m^{l-}_{j\text{RetProd}} - \overline{m}_{ik} \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.92)$$

$$\leq \overline{w}^{\circ k'}_{ik} \sum_{j \in \mathcal{S}_k} m^{l-}_{j\text{RetProd}} + \overline{m}_{ik} \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.93)$$

for all $i \in \mathcal{E}$, $k \in \mathcal{Z}_{\text{NGProd}}$, and $k' \in \mathcal{T}_{\text{PSC}}$,

$$\sum_{j \in \mathcal{S}_k} w_{ij} m^l_{j\text{Prod}} \geq \underline{w}^{\text{Blow}k'}_{ik} \sum_{j \in \mathcal{S}_k} m^l_{j\text{Prod}} - \overline{m}_{ik} \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.94)$$

$$\leq \overline{w}^{\text{Blow}k'}_{ik} \sum_{j \in \mathcal{S}_k} m^l_{j\text{Prod}} + \overline{m}_{ik} \left(1 - \beta^l_{\text{Type}k'}\right) \quad (4.95)$$

for all $i \in \mathcal{E}$, $k \in \mathcal{Z}_{\text{NGProd}}$, and $k' \in \mathcal{T}_{\text{PSC}}$,

$$m^{l-}_{j\text{RetProd}} \geq \underline{w}^{\circ k'}_{jk} \sum_{j' \in \mathcal{S}_k} m^{l-}_{j'\text{RetProd}} - \overline{m}_{j\text{Prod}} \left(1 - \beta^{l}_{\text{Type}k'}\right) \tag{4.96}$$

$$\leq \overline{w}^{\circ k'}_{jk} \sum_{j' \in \mathcal{S}_k} m^{l-}_{j'\text{RetProd}} + \overline{m}_{j\text{Prod}} \left(1 - \beta^{l}_{\text{Type}k'}\right) \tag{4.97}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$, $j \in \mathcal{S}_k$, and $k' \in \mathcal{T}_{\text{PSC}}$,

$$m^{l}_{j\text{Prod}} \geq \underline{w}^{\text{Blow}k'}_{jk} \sum_{j' \in \mathcal{S}_k} m^{l}_{j'\text{Prod}} - \overline{m}_{j\text{Prod}} \left(1 - \beta^{l}_{\text{Type}k'}\right) \tag{4.98}$$

$$\leq \overline{w}^{\text{Blow}k'}_{jk} \sum_{j' \in \mathcal{S}_k} m^{l}_{j'\text{Prod}} + \overline{m}_{j\text{Prod}} \left(1 - \beta^{l}_{\text{Type}k'}\right) \tag{4.99}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$, $j \in \mathcal{S}_k$, and $k' \in \mathcal{T}_{\text{PSC}}$. Equations 4.80-91 employ upper bounds for stream masses,

$$\overline{m}_k = \begin{cases} \rho_k \max\limits_{j \in \{1,\ldots,n_{\text{PSC}}\}} \overline{v}^{\text{PSC}j} & \text{if } k \in \mathcal{Z}_{\text{NGFeed}} \\ \\ \max\limits_{j \in \mathcal{S}_k} \overline{m}_{j\text{Prod}} & \text{if } k \in \mathcal{Z}_{\text{NGProd}} \end{cases}$$

in which $\overline{m}_{j\text{Prod}}$ is computed as for Equation 4.42. The mass upper bound in of Equations 4.92-95 can be taken as

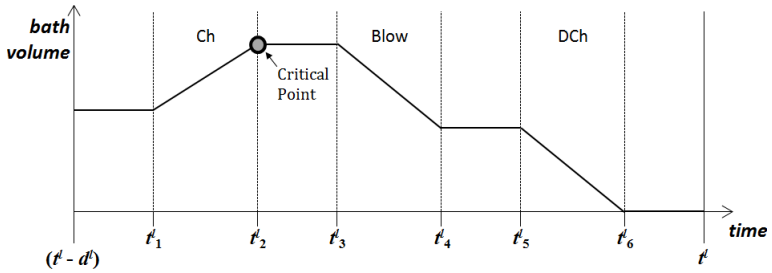
$$\overline{m}_{ik} = \max_{j \in \mathcal{S}_k} \left(w_{ij} \overline{m}_{j\text{Prod}}\right)$$

for all $i \in \mathcal{E}$ and $k \in \mathcal{Z}_{\text{NGProd}}$.

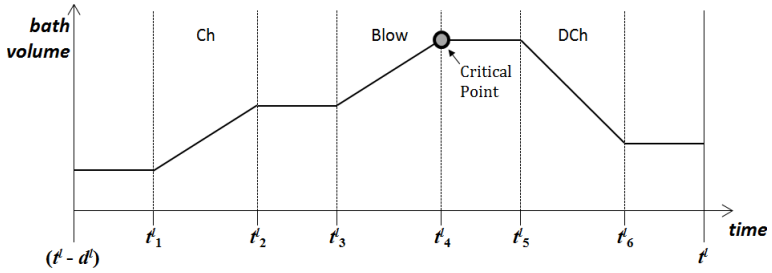
Equations 4.74-99 impose upper and lower bounds for the bath composition. More complicated composition constraints may require the direct implementation of mass-fractions, which falls outside of the MILP framework, as discussed in Section 6.1.

4.5.3 Volume Constraints

To address the possibility of bath overflow, Figure 4.3 identifies two critical times during the converter transitions. As shown in Figure 4.3a, the maximum volume may be attained at the end of charging; this is true of transitions that do not include a Slag-Blow, but may include a Copper-Blow or various forms of overblowing. According to Figure 4.3b, the maximum volume may also be attained at the end of blowing, particularly in transitions that include the Slag-Blow reaction, as discussed in Subsection 1.3.2.



(a) Critical point at the end of charging



(b) Critical point at the end of blowing

Figure 4.3: Volume evolution during converter transition

To prevent overflows from occurring during charging,

$$\sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} \frac{m^{l-}_{\mathrm{Ret}k}}{\rho_k} + \sum_{j\in\mathcal{S}_{\mathrm{NGProd}}} \frac{m^{l-}_{j\mathrm{RetProd}}}{\rho_j} + \sum_{j\in\mathcal{F}_{\mathrm{Ch}}} v^{jl} \leq \overline{v}^{\mathrm{obj}(l)} - \sum_{k\in\mathcal{T}_{\mathrm{PSC}}} \left(\overline{v}^{\mathrm{obj}(l)} - \overline{v}^{\mathrm{Ch}k,\mathrm{obj}(l)}\right) \beta^l_{\mathrm{Type}k} \tag{4.100}$$

and during blowing,

$$\sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} \frac{m^{l}_{\mathrm{Ret}k}}{\rho_k} + \sum_{j\in\mathcal{S}_{\mathrm{NGProd}}} \frac{m^{l}_{j\mathrm{Prod}}}{\rho_j} \leq \overline{v}^{\mathrm{obj}(l)} - \sum_{k\in\mathcal{T}_{\mathrm{PSC}}} \left(\overline{v}^{\mathrm{obj}(l)} - \overline{v}^{\mathrm{Blow}k,\mathrm{obj}(l)}\right) \beta^l_{\mathrm{Type}k} \tag{4.101}$$

for all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$. The parameters $\overline{v}^{\mathrm{Ch}k,\mathrm{obj}(l)}$ and $\overline{v}^{\mathrm{Blow}k,\mathrm{obj}(l)}$ are the maximum allowable bath volume within a converter $\mathrm{obj}(l)$ following charging and following blowing, respectively, during a transition of type $k \in \mathcal{T}_{\mathrm{PSC}}$; such parameters are related to the maximum allowable volume in the converter,

$$\overline{v}^{\mathrm{PSC}j} \geq \max\left(\max_{k\in\mathcal{T}_{\mathrm{PSC}}} \overline{v}^{\mathrm{Ch}k,\mathrm{PSC}j}, \max_{k\in\mathcal{T}_{\mathrm{PSC}}} \overline{v}^{\mathrm{Blow}k,\mathrm{PSC}j}\right)$$

for $j \in \{1, 2, \ldots, n_{\mathrm{PSC}}\}$. Indeed, $\overline{v}^{\mathrm{PSC}j}$ is an appropriate default value for $\overline{v}^{\mathrm{Ch}k,\mathrm{PSC}j}$ and $\overline{v}^{\mathrm{Blow}k,\mathrm{PSC}j}$.

Equations 4.100-101 each include contributions from both the feed and the product streams on the lefthand side. Additionally, Equation 4.101 includes the newly charged volume.

Only the essential volumetric constraints have been described. Other examples may include restrictions relating the volumes of the different product streams (e.g. there should never be more than twice the volume of slag as matte, etc.), or there may be restrictions on the solid versus liquid charges. It is a relatively simple matter to incorporate these additional restrictions into the MILP, if need be.

4.5.4 Temperature Constraints

The Peirce-Smith reactions require that the bath temperature remain within certain limits that must be respected throughout the blowing actions. There is a danger that too much cold charge might be added, which overcools the bath and causes kinetic difficulties during the subsequent blow. Since the reactions are exothermic, there is also a danger of overheating, especially toward the end of the blow. Thus, critical temperatures are observed at the beginning and end of the blow (Figure 4.4). Corresponding temperature bounds, $\underline{T}^{\text{Blow}k}$ and $\overline{T}^{\text{Blow}k}$, are prescribed for every transition type $k \in \mathcal{T}_{\text{PSC}}$.

Following the treatment of Subsections 4.3.6 and 4.4.3, $\underline{T}^{\text{DCh}k}$ and $\overline{T}^{\text{DCh}k}$ are imposed the discharge transitions $k \in \mathcal{T}_{\text{PSC}}$. For any transition type k that includes an intermediate discharge, the MILP considers only a single, fixed value $T^{\text{DCh}k}$; this is to say that $\underline{T}^{\text{DCh}k} = \overline{T}^{\text{DCh}k} = T^{\text{DCh}k}$ for all $k \in \mathcal{T}_{\text{PSC,IDCh}}$. The possibility of variable skimming temperatures is discussed in Section 6.1.

Bath temperatures cannot be directly incorporated into the MILP as variables, because they have a nonlinear relationship to the heat and mass variables. Nonetheless, it is possible to implement upper and lower bounds on temperatures, in an indirect manner, as discussed

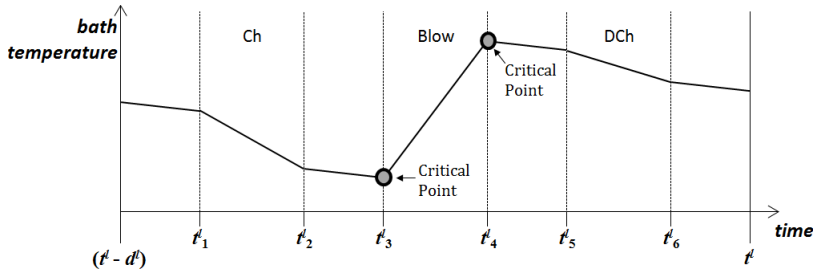


Figure 4.4: Temperature evolution during converter transition

in Subsections 3.2.3 and 3.2.4. The approach is based on an adaptation of Equation 3.19,

$$h^{\mathrm{obj}(l)}(t) = \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(T^{\mathrm{obj}(l)}(t), t) \right] \left[m_{k'}^{\mathrm{obj}(l)}(t) \right]$$

which expresses the heat content of the bath in obj(l) as a function of time t, in which $m_{k'}^{\mathrm{obj}(l)}(t)$ is the mass of k' in the bath of obj(l).

Converter temperature bounds were introduced in Subsection 4.2.1, such that $\underline{T}^{\mathrm{obj}(l)} \leq T^{\mathrm{obj}(l)}(t) \leq \overline{T}^{\mathrm{obj}(l)}$, where $T^{\mathrm{obj}(l)}(t)$ is the bath temperature in obj(l) at time t. Considering that the beginning of a blow t_3^l corresponds to the coldest blowing time, and that the end of a blow t_4^l corresponds to the hottest blowing time (Figure 4.4), it follows that $T^{\mathrm{obj}(l)}(t_3^l) \geq \underline{T}^{\mathrm{obj}(l)}$ and $T^{\mathrm{obj}(l)}(t_4^l) \leq \overline{T}^{\mathrm{obj}(l)}$. In terms of heat,

$$h(t_3^l) \geq \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(\underline{T}^{\mathrm{obj}(l)}, t_3^l) \right] \left[m_{k'}^{\mathrm{obj}(l)}(t_3^l) \right]$$

$$h(t_4^l) \leq \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(\overline{T}^{\mathrm{obj}(l)}, t_4^l) \right] \left[m_{k'}^{\mathrm{obj}(l)}(t_4^l) \right]$$

The first inequality is satisfied if and only if $T^{\mathrm{obj}(l)}(t_3^l) \geq \underline{T}^{\mathrm{obj}(l)}$, and the second inequality is satisfied if and only if $T^{\mathrm{obj}(l)}(t_4^l) \leq \overline{T}^{\mathrm{obj}(l)}$; this equivalence is related to the positivity of the specific heat capacity, as discussed in Subsection 3.2.3.

The MILP considers that different blowing transitions may have different bounds. Binary terms are therefore appended the previous inequalities,

$$h^{\mathrm{obj}(l)}(t_3^l) \geq \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(\underline{T}^{\mathrm{Blow}k}, t_3^l) \right] \left[m_{k'}^{\mathrm{obj}(l)}(t_3^l) \right] - (\overline{h}^{\mathrm{obj}(l)} - \underline{h}^{\mathrm{obj}(l)}) \left(1 - \beta_{\mathrm{Type}k}^l \right)$$

$$h^{\mathrm{obj}(l)}(t_4^l) \leq \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(\overline{T}^{\mathrm{Blow}k}, t_4^l) \right] \left[m_{k'}^{\mathrm{obj}(l)}(t_4^l) \right] + (\overline{h}^{\mathrm{obj}(l)} - \underline{h}^{\mathrm{obj}(l)}) \left(1 - \beta_{\mathrm{Type}k}^l \right)$$

for all k in $\mathcal{T}_{\mathrm{PSC}}$, and all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$. Thus if l is a blowing transition of type k, then the binary terms disappear and it follows ultimately that $\underline{T}^{\mathrm{Blow}k} \leq T^{\mathrm{obj}(l)}(t) \leq \overline{T}^{\mathrm{Blow}k}$ for all times t from t_3^l to t_4^l. The difference $(\overline{h}^{\mathrm{obj}(l)} - \underline{h}^{\mathrm{obj}(l)})$ ensures that the inequalities are slack if l is not of type k.

The first inequality, $T(t_3^l) \geq \underline{T}^{\text{Blow}k}$, is incorporated into the MILP as

$$h_{\text{Ret}}^{l-} + h_{\text{Ch}}^{l} - \sum_{i=0}^{3} h_{\text{Env}i}^{l} \;\geq\; \sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} \underline{w}_{\text{H}k'}^{\text{Blow}k} m_{\text{Ret}k'}^{l-} \;+\; \sum_{j \in \mathcal{S}_{\text{NGProd}}} \underline{w}_{\text{H}j}^{\text{Blow}k} m_{j\text{RetProd}}^{l-}$$
$$+ \sum_{j \in \mathcal{F}_{\text{Ch}}} \underline{w}_{\text{H,srce}(j)}^{\text{Blow}k} \rho_{\text{srce}(j)} v^{jl} \;-\; (\overline{h}^{\text{obj}(l)} - \underline{h}^{\text{obj}(l)}) \left(1 - \beta_{\text{Type}k}^{l}\right) \tag{4.102}$$

for all k in $\mathcal{T}_{\text{PSC}}$ and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$, in which $\underline{w}_{\text{H}j}^{\text{Blow}k} = w_{\text{H}j}(\underline{T}^{\text{Blow}k})$ and $\underline{w}_{\text{H,srce}(j)}^{\text{Blow}k} = w_{\text{H,srce}(j)}(\underline{T}^{\text{Blow}k})$. On the lefthand side, $h(t_3^l)$ includes the retained heat from the previous transition, plus the heat of the newly charged feed, minus the environmental losses. On the righthand side, the summation $\sum_{k' \in \mathcal{Z}_{\text{NG}}}$ is decomposed into the previously retained feeds and products, plus the newly charged material.

The second inequality, $T(t_4^l) \leq \overline{T}^{\text{Blow}k}$, is implemented as

$$h_{\text{Ret}}^{l-} + h_{\text{Ch}}^{l} + h_{\text{NGBlow}}^{l} + h_{\text{Blast}}^{l} - h_{\text{Offgas}}^{l} - \sum_{i=0}^{4} h_{\text{Env}i}^{l}$$
$$\leq \tag{4.103}$$
$$\sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} \overline{w}_{\text{H}k'}^{\text{Blow}k} m_{\text{Ret}k'}^{l} + \sum_{j \in \mathcal{S}_{\text{NGProd}}} \overline{w}_{\text{H}j}^{\text{Blow}k} m_{j\text{Prod}}^{l} + (\overline{h}^{\text{obj}(l)} - \underline{h}^{\text{obj}(l)}) \left(1 - \beta_{\text{Type}k}^{l}\right)$$

for all k in $\mathcal{T}_{\text{PSC}}$, and all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$, in which $\overline{w}_{\text{H}j}^{\text{Blow}k} = w_{\text{H}j}(\overline{T}^{\text{Blow}k})$ and $\overline{w}_{\text{H,srce}(j)}^{\text{Blow}k} = w_{\text{H,srce}(j)}(\overline{T}^{\text{Blow}k})$. On the lefthand side, $h(t_4^l)$ includes the terms of $h(t_3^l)$, as well as the accumulated contribution of the blowing action, $h_{\text{NGBlow}}^{l} + h_{\text{Blast}}^{l} - h_{\text{Offgas}}^{l} - h_{\text{Env4}}^{l}$. On the righthand side, there are again contributions from both the feed and product streams.

The discharge temperature bounds can be stated in terms of temperature $\underline{T}^{\text{DCh}k} \leq T^{\text{obj}(l)}(t_6^l) \leq \overline{T}^{\text{DCh}k}$, for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}$. Following the same approach as for Equations 4.102-103,

$$h(t_6^l) \;\geq\; \sum_{k' \in \mathcal{Z}_{\text{NG}}} \left[w_{\text{H}k'}(\underline{T}^{\text{DCh}k}, t_6^l)\right] \left[m_{k'}^{\text{obj}(l)}(t_6^l)\right] - (\overline{h}^{\text{obj}(l)} - \underline{h}^{\text{obj}(l)}) \left(1 - \beta_{\text{Type}k}^{l}\right)$$
$$\leq \sum_{k' \in \mathcal{Z}_{\text{NG}}} \left[w_{\text{H}k'}(\overline{T}^{\text{DCh}k}, t_6^l)\right] \left[m_{k'}^{\text{obj}(l)}(t_6^l)\right] + (\overline{h}^{\text{obj}(l)} - \underline{h}^{\text{obj}(l)}) \left(1 - \beta_{\text{Type}k}^{l}\right)$$

Equivalently,

$$h^{l-}_{\mathrm{Ret}} + h^{l}_{\mathrm{Ch}} + h^{l}_{\mathrm{NGBlow}} + h^{l}_{\mathrm{Blast}} - h^{l}_{\mathrm{Offgas}} - \sum_{i=0}^{5} h^{l}_{\mathrm{Env}i}$$

$$\geq \tag{4.104}$$

$$\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} \underline{w}^{\mathrm{DCh}k}_{\mathrm{H}k'} m^{l}_{\mathrm{Ret}k} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \underline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} m^{l}_{j\mathrm{RetProd}} - (\overline{h}^{\mathrm{obj}(l)} - \underline{h}^{\mathrm{obj}(l)}) \left(1 - \beta^{l}_{\mathrm{Type}k}\right)$$

and

$$h^{l-}_{\mathrm{Ret}} + h^{l}_{\mathrm{Ch}} + h^{l}_{\mathrm{NGBlow}} + h^{l}_{\mathrm{Blast}} - h^{l}_{\mathrm{Offgas}} - \sum_{i=0}^{5} h^{l}_{\mathrm{Env}i}$$

$$\leq \tag{4.105}$$

$$\sum_{k' \in \mathcal{Z}_{\mathrm{NGFeed}}} \overline{w}^{\mathrm{DCh}k}_{\mathrm{H}k'} m^{l}_{\mathrm{Ret}k} + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \overline{w}^{\mathrm{DCh}k}_{\mathrm{H}j} m^{l}_{j\mathrm{RetProd}} + (\overline{h}^{\mathrm{obj}(l)} - \underline{h}^{\mathrm{obj}(l)}) \left(1 - \beta^{l}_{\mathrm{Type}k}\right)$$

for all k in $\mathcal{T}_{\mathrm{PSC}}$, and all $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$. The lefthand side of Equations 4.104-105 includes an additional environmental heat loss terms, $h^{l}_{\mathrm{Env}5}$, which had not been present in Equation 4.103.

The implementation of temperature bounds (Equations 4.102-105) relies on the heat bounds, $\underline{h}^{\mathrm{PSC}j}$ and $\overline{h}^{\mathrm{PSC}j}$ that had been introduced in Subsection 4.2.1, and which had been constructed using converter temperature bounds, $\underline{T}^{\mathrm{PSC}j}$ and $\overline{T}^{\mathrm{PSC}j}$. These temperature bounds should satisfy

$$\underline{T}^{\mathrm{PSC}j} \leq \min\left(T^{\mathrm{Offgas}}, \min_{j' \in \mathcal{F}_{\mathrm{Feed}}} T^{j'}, \min_{k \in \mathcal{T}_{\mathrm{PSC}}} \underline{T}^{\mathrm{Blow}k}, \min_{k \in \mathcal{T}_{\mathrm{PSC}}} \underline{T}^{\mathrm{DCh}k}\right)$$

$$\overline{T}^{\mathrm{PSC}j} \geq \max\left(T^{\mathrm{Offgas}}, \max_{j' \in \mathcal{F}_{\mathrm{Feed}}} T^{j'}, \max_{k \in \mathcal{T}_{\mathrm{PSC}}} \overline{T}^{\mathrm{Blow}k}, \max_{k \in \mathcal{T}_{\mathrm{PSC}}} \overline{T}^{\mathrm{DCh}k}\right)$$

for $j \in \{1, 2, \ldots, n_{\mathrm{PSC}}\}$. Incidentally, $\min_{j \in \{1,2,\ldots,n_{\mathrm{PSC}}\}} \underline{T}^{\mathrm{PSC}j}$ can help provide appropriate default values for $\underline{T}^{\mathrm{Blow}k}$ and $\underline{T}^{\mathrm{DCh}k}$. Likewise, $\max_{j \in \{1,2,\ldots,n_{\mathrm{PSC}}\}} \overline{T}^{\mathrm{PSC}j}$ can help provide default values for $\overline{T}^{\mathrm{Blow}k}$ and $\overline{T}^{\mathrm{DCh}k}$.

Equations 4.102-105 impose bounds for the bath temperature. In some situations it may be necessary to impose temperature bounds that differ from converter to converter, e.g.

$\underline{T}^{\text{Blow}k,\text{PSC}j} \leq T^{\text{PSC}j} \leq \overline{T}^{\text{Blow}k,\text{PSC}j}$ rather than $\underline{T}^{\text{Blow}k} \leq T^{\text{PSC}j} \leq \overline{T}^{\text{Blow}k}$, which would require a slight modification of Equations 4.102-105. More complicated thermal constraints may require the direct implementation of temperature as a function of heat, but this falls outside of the MILP framework, as discussed in Section 6.1.

4.5.5 Indirect Transition Constraints in General Linear Form

$\mathcal{L}_{\text{PSC}}^{\text{Trans}}$ is the set of transition feasibility clauses that are implemented in linear form. These clauses include coefficients comparable to those of Equation 4.73 for the direct transition constraints, but there are now additional terms that correspond to the essential intermediate variables for the preceding transition $l-$ and the current transition l.

Considering only the essential intermediate variables described previously,

$$\begin{aligned}
&\sum_{k\in\mathcal{Z}_{\text{NGFeed}}} a^i_{m\text{Ret-},k} m^{l-}_{\text{Ret}k} \;+\; \sum_{k\in\mathcal{S}_{\text{NGProd}}} a^i_{m,j\text{RetProd-}} m^{l-}_{j\text{RetProd}} \;+\; a^i_{h\text{Ret-}} h^{l-}_{\text{Ret}} \\
+\; &\sum_{k\in\mathcal{T}_{\text{PSC}}} a^i_{\beta\text{-},k} \beta^{l-}_{\text{Type}k} \;+\; \sum_{i'=0}^{7} a^i_{d,i'} d^l_i \;+\; \sum_{j\in\mathcal{F}_{\text{NG}}} a^i_{v,j} v^{jl} \\
+\; &\sum_{j\in\mathcal{F}^{\text{MSM}}} a^i_{u,j} u^{jl} \;+\; \sum_{k\in\mathcal{T}_{\text{PSC}}} a^i_{\beta,k} \beta^{l}_{\text{Type}k} \;+\; \sum_{i'\in\mathcal{E}} a^i_{m,i'\text{Blast-}} m^{l-}_{i'\text{Blast}} \\
+\; &\sum_{j\in\mathcal{S}_{\text{Prod}}} a^i_{m\text{Prod-},j} m^{l-}_{j\text{Prod}} \;+\; a^i_{h\text{Blast-}} h^{l-}_{\text{Blast}} \;+\; a^i_{h\text{Offgas-}} h^{l-}_{\text{Offgas}} \\
+\; &a^i_{h\text{DCh-}} h^{l-}_{\text{DCh}} \;+\; \sum_{i'=0}^{7} a^i_{h\text{Env-},i'} h^{l-}_{\text{Env}i'} \;+\; \sum_{i'\in\mathcal{E}} a^i_{m,i'\text{Blast}} m^{l}_{i'\text{Blast}} \\
+\; &\sum_{j\in\mathcal{S}_{\text{Prod}}} a^i_{m\text{Prod},j} m^{l}_{j\text{Prod}} \;+\; a^i_{h\text{Blast}} h^{l}_{\text{Blast}} \;+\; a^i_{h\text{Offgas}} h^{l}_{\text{Offgas}} \\
+\; &a^i_{h\text{DCh}} h^{l}_{\text{DCh}} \;+\; \sum_{i'=0}^{7} a^i_{h\text{Env},i'} h^{l}_{\text{Env}i'} \;\leq\; b^i
\end{aligned} \tag{4.106}$$

for all $i \in \mathcal{L}_{\text{PSC}}^{\text{Trans}}$ and all $l \in \mathcal{A}_{\text{PSC}} \backslash \mathcal{A}_{\circ}$. Depending on the problem at hand, it may be necessary implement more essential intermediate variables. By comparing Equation 4.106 to 4.73, any element of $\mathcal{L}_{\text{PSC}}^{\text{DTrans}}$ can be implemented as an element of $\mathcal{L}_{\text{PSC}}^{\text{Trans}}$, but that the converse is not generally true.

The composition, volume and temperature constraints (Equations 4.74-105) fall into the framework of Equation 4.106. Nonetheless, they have been treated separately for instructive purposes. As demonstrated in Section 5.3, Equation 4.106 can be also used to implement overblow conditions.

Equation 4.106 is general, and leaves the possibility of customizing special constraints, which include any linear combination of intermediate variables, preceding state variables and current transition variables.

4.6 Global Objectives and Constraints

4.6.1 Optimization of Nongaseous Flows and of Transition Types

Instances of the PSC Problem usually fit one of the following two scenarios [7],

- Maximize production, without over- or under-consuming any of the limited resources
- Maximize or minimize the consumption of a limited resource, without over- or under-consuming any of the other limited resources, and while exceeding a prescribed level of production.

Therefore the optimization objective can be the maximization of production, or it can be the maximization or minimization of the consumption of a certain resource.

There are several ways of measuring the production of a given schedule. The particular measure depends on the nature of the problem, but is usually expressed as a sum over all of the converter transitions in the current schedule. For example,

$$\max f = \sum_{l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ} \left(\sum_{j \in \mathcal{F}_k} v^{jl} \right)$$

can be to maximize the intake of feed matte if $k =$ FMatte. Alternatively, it can be to maximize the discharging of the main product, be it $k =$ Blister for a copper smelter or $k =$ CMatte for a nickel-copper smelter. It is also conceivable that the objective would be

$$\min f = \sum_{l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_\circ} \left(\sum_{j \in \mathcal{F}_k} v^{jl} \right)$$

hence to minimize the consumption or production of a certain stream, e.g. if k is a certain undesirable type of slag.

Under slightly different circumstances, the main objective may be to maximize or minimize the occurrence of certain transition types. For example, the objective may be to minimize the amount of recharging, (while maintaining a prescribed level of production, perhaps). In this case the objective would be

$$\min f = \sum_{l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}} \left(\sum_{k \in \mathcal{T}_{\mathrm{PSC,Avoid}}} \beta^{l}_{\mathrm{Type}k} \right)$$

where $\mathcal{T}_{\mathrm{PSC,Avoid}} \subset \mathcal{T}_{\mathrm{PSC}}$ is the subset of transition types that is to be avoided.

The following general form has been implemented into the current MILP formulation.

$$\max f = \sum_{l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}} \left(\sum_{j \in \mathcal{F}_{\mathrm{NG}}} \chi_{v,j} v^{jl} + \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} \chi_{\beta,k} \beta^{l}_{\mathrm{Type}k} \right) \tag{4.107}$$

in which the coefficients, $\chi_{v,j}$ and $\chi_{\beta,k}$, determine the relative importance of the objective variables. Traditionally, the letter c is used instead of χ to denote the objective weighting (Appendix B); in the PSC Problem, however, c is reserved for the specific heat capacity (Chapter 3). As discussed in Appendix B.1, a minimization can be converted into a maximization simply by reversing the signs of the coefficients.

Equation 4.107 is representative of industrial objectives [7], considering both the nongaseous flows and the transition types. Moreover, this objective can be extended to include any of the other variables described in Sections 4.2 and 4.3.

4.6.2 Limiting of Nongaseous Flows and of Transition Types

Global constraints are applied over the entire schedule, and possibly into the next schedule, to control production and resource consumption. Thus, there is a certain interplay between these global constraints and the objective.

Flows are commonly subject to lower and upper bounds. To clear inventory space, it may be imperative to consume a certain amount of a reverts stream, for example. Alternatively, there may be only a limited amount of copper scrap available for the current schedule which, if improperly managed, will lead to overheating during the Copper-Blow stage. There may be upper or lower limits on the application of certain transition types, due to scarcity of human resources or technology. For example, it may not be practical to perform more than three scrap deliveries in the current schedule.

The set of linear global constraints is denoted $\mathcal{L}_{\text{PSC}}^{\text{Global}}$. Through similar considerations as in the previous subsection, these constraints have been implemented as

$$\sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\circ}} \left(\sum_{j \in \mathcal{F}_{\text{NG}}} a_{v,j}^{i} v^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} a_{\beta,k}^{i} \beta_{\text{Type}k}^{l} \right) \leq b^{i} \tag{4.108}$$

for all $i \in \mathcal{L}_{\text{PSC}}^{\text{Global}}$. Each clause i is characterized by the coefficients $(a_{v,j}^{i},\ a_{\beta,k}^{i})$ and the righthand constant b^{i}. To include "$\geq$" constraints, it is sufficient to reverse the signs of the parameters, as described in Appendix B.1. Global equalities can be implemented by combining "$\geq$" and "$\leq$" constraints.

The framework developed in this chapter is rather general and applies to numerous problems that are encountered in copper and nickel smelters. However, each of these problems requires some adaptation in specifying the parameters, and possibly the creation of customized constraints and variables. This adaptation process is demonstrated in Chapter 5.

CHAPTER 5

THE SINGLE-CYCLE PSC PROBLEM AND SAMPLE COMPUTATIONS

5.1 Adaptation of the PS MILP Formulation

5.1.1 Topological and Initial Conditions

In current practice, operational decisions are often made on a cycle-to-cycle basis [47, 66]. As described in Subsection 1.3.3, a PS converting cycle begins by charging an empty converter with several ladles of matte, and a combination of flux and secondary feed. Following a sequence of blowing and recharging, the cycle terminates when the main product is withdrawn, leaving an empty converter, ready for the next cycle.

The formulation developed in Chapters 2-4 considers the simultaneous operation of several converters, over several cycles. This general formulation can be parametrized so as to consider a single cycle. Firstly n_{PSC} is set to 1, thereby focusing on a single converter. Secondly, the feasible solutions should describe the evolution of the converter as an open path, connecting the initial empty state to the final empty state (Figure 5.1). This variant of the PSC Problem is referred to as the Single Cycle Peirce-Smith Converter (SC-PSC) Problem.

In accordance with Figure 5.1, the SC-PSC Problem requires that $\mathcal{T}_{\text{PSC,Empty}}$ contain a transition type EndPreviousCycle, such that

$$\mathcal{T}^{-}_{\text{PSC,EndPreviousCycle}} = \varnothing \tag{5.1}$$

and a transition type EndCurrentCycle, such that

$$\text{EndCurrentCycle} \notin \mathcal{T}^{-}_{\text{PSC}k} \tag{5.2}$$

for all $k \in \mathcal{T}_{\text{PSC}}$.

Figure 5.2 is similar to Figure 2.4, except that the transition type EndCycle has been

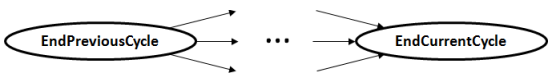


Figure 5.1: EndPreviousCycle and EndCurrentCycle transitions

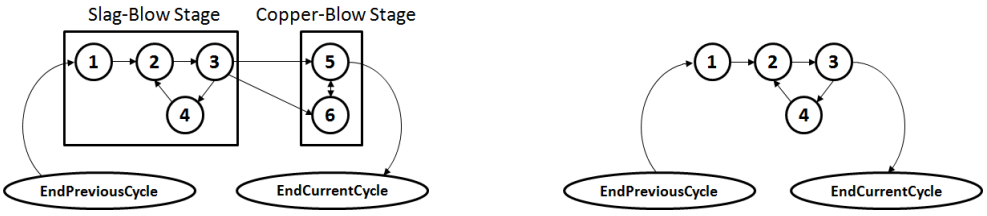


Figure 5.2: Converter transition diagrams to adapt (a) typical copper PS systems, and (b) typical nickel PS System to the SC-PSC formulation. The transition types are numbered (1) InitialCharge, (2) SlagBlow, (3) Skim, (4) Recharge, (5) CopperBlow, (6) ScrapCharge, and (7) EndCycle.

replaced by EndPreviousCycle and EndCurrentCycle, to fulfill the requirements of the SC-PSC.

In the SC-PSC Problem, it is often convenient to measure time from the beginning of the current cycle $t_{\text{Begin}} = 0$. Also, the current cycle is distinct from the next cycle, so that $t_{\text{End}} = \bar{t}$. For a typical application, the SC-PSC may thus be confined to a time interval from 0 to $\bar{t}$.

After the previous cycle that has ended at $t^{(\text{PSC},1,0)} \leq t_{\text{Begin}}$, the converter is left empty. Therefore the initial conditions for the current cycle are described as,

$$m_{\text{Ret}k}^{(\text{PSC},1,0)} = 0 \tag{5.3}$$

for all $i \in \mathcal{Z}_{\text{NGFeed}}$,

$$m_{j\text{RetProd}}^{(\text{PSC},1,0)} = 0 \tag{5.4}$$

for all $k \in \mathcal{S}_{\text{NGProd}}$,

$$h_{\text{Ret}}^{(\text{PSC},1,0)} = 0 \tag{5.5}$$

and

$$\text{Type}^{(\text{PSC},1,0)} = \text{EndPreviousCycle}$$

This last expression is implemented in the MILP using the transition-type determinants,

$$\beta_{\text{Type}k}^{(\text{PSC},1,0)} = \begin{cases} 1 & \text{if } k = \text{EndPreviousCycle} \\ 0 & \text{if } k \in \mathcal{T}_{\text{PSC}} \setminus \{\text{EndPreviousCycle}\} \end{cases} \tag{5.6}$$

Equations 5.3-6 impose initial conditions on $(\text{PSC}, 1, 0)$ so that the successor $(\text{PSC}, 1, 1)$

becomes the first transition of the current cycle. Incidentally, Equation 5.3 also implies that $m_{j_{\text{RetProd}}}^{(\text{PSC},1,0)} = 0$ for all $j \in \mathcal{S}_{\text{NGProd}}$.

The following global equality must be appended to the MILP formulation so that feasible solutions contain exactly one complete converting cycle,

$$\sum_{k=1}^{\overline{n}_{\text{Asgn,PSC}}} \beta_{\text{Type,EndCurrentCycle}}^{(\text{PSC},1,k)} = 1 \tag{5.7}$$

According to Equation 5.7, each feasible solution is associated with a number $n_{\text{Asgn,PSC}} \in \{1, 2, \ldots, \overline{n}_{\text{Asgn,PSC}}\}$, such that $\text{Type}^{(\text{PSC},1,n_{\text{Asgn,PSC}})} = \text{EndCurrentCycle}$. Following the SC-PSC structure, $(\text{PSC}, 1, n_{\text{Asgn,PSC}})$ must be the final transition because EndCurrentCycle does not lead into any subsequent transition types; $n_{\text{Asgn,PSC}}$ is therefore the number of converter assignments (transitions) within the current cycle.

Early computational efforts for the SC-PSC Problem have been successful, even using a standard retail computer (Toshiba Qosmio©, with Intel Core i7 CPU, [70]), with a standard MILP solver platform (CPLEX, [71, 72]). As demonstrated in Sections 5.3 and 5.4, this has allowed the verification of the forward state computation as well as the transition feasibility conditions.

The MILP formulation should also work correctly for general instances of the PSC Problem, although a more extensive computational approach seems to be required; this may involve parallel computing [73]. Current research is devoted to specialized algorithms that take advantage of the particular structure of the PSC Problem, as discussed in Section 6.2.

5.1.2 Critical Overlap Decomposition

The Critical Overlap (CO) decomposition is a means of adapting the SC-PSC Problem for the management of the converting aisle. This adaptation does not generally guarantee optimality as would the general MILP formulation. Nonetheless, it uses the innate structure of the PSC Problem, and is a fair representation of current scheduling practice [47, 75, 76].

The CO decomposition contends that there is no more than one critical stage during a cycle that suffers from a shortage of ancillary objects (Figure 5.3). There is hence an abundance of objects to assist the precritical stage, and to assist the postcritical stage.

At any time in the schedule, n_{Crit} denotes the number of converters that are in the critical stage. The CO formulation asserts that n_{Crit} is subject to an upper bound $\overline{n}_{\text{Crit}}$, which is

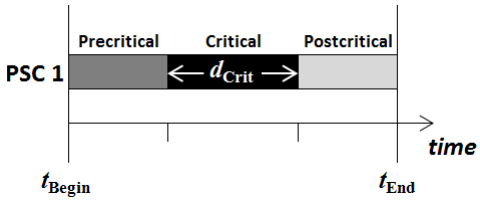


Figure 5.3: Critical Overlap Decomposition

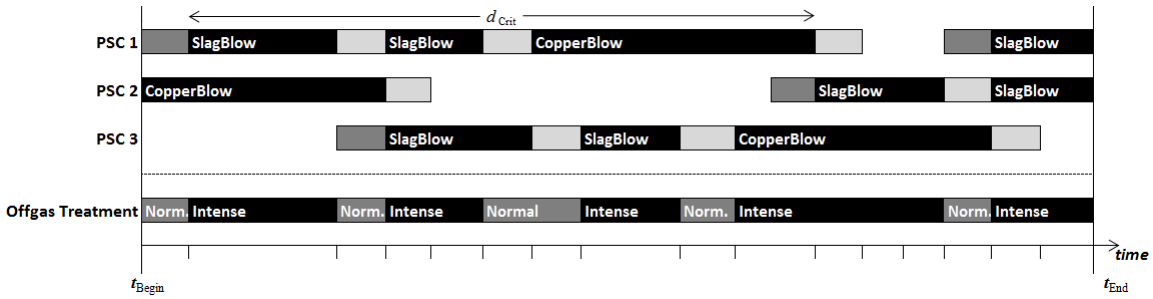


Figure 5.4: Offgas treatment capacity limiting production to no more than two simultaneous blowing actions

presumably less than n_{PSC}. Henceforth, it will be assumed that $\overline{n}_{\text{Crit}} \leq n_{\text{PSC}} - 1$.

An appropriate upper bound $\overline{n}_{\text{Crit}}$ can be deduced by considering the dependency clauses, $\mathcal{D}_{ik}$ for $i \in \mathcal{C}$ and $k \in \mathcal{T}_i$. Effectively, the CO decomposition replaces the dependency clauses $\mathcal{D}_{ik}$ with a single number $\overline{n}_{\text{Crit}}$.

Figure 5.4 illustrates how the CO decomposition is used to construct Gantt charts when there is only one cycle design under consideration. A greedy approach is used to "stuff" as many cycles as possible within the schedule [74], while respecting the limitation that there may never be more than $\overline{n}_{\text{Crit}}$ overlapping critical stages.

In the case of Figure 5.4, $\overline{n}_{\text{Crit}} = 2$ is a limitation imposed by the offgas treatment facilities, as is often the case [47, 66, 75, 76]. Within each cycle, the critical stage begins with the first Slag-Blow and extends until the end of the Copper-Blow. Within the first half of the schedule in Figure 5.4, the end of a critical stage in PSC 2 coincides with the beginning of a critical stage in PSC 3. Later on, the end of a critical stage in PSC 1 coincides with the beginning of a critical stage in PSC 2, and the end of a critical stage in PSC 3 coincides with the beginning of a critical stage in PSC 1.

The converters are programmed in a staggered manner in Figure 5.4, so as to manage the (abundant) ancillary objects that are associated with pre- and post-critical stages. This

staggered greedy approach is most appropriate when the cycles are dominated by the critical stage, as discussed in Subsection 5.1.3.

The Chuquicamata smelter applies three kinds of converting cycles, and the 24 hour production schedule is constructed through an exhaustive enumeration of possible schedules [47]; in this case, there is a relatively small number possibilities since each of the cycles last roughly 7 hours, and there are at most two simultaneous blowing operations at any given time. This is quite different from the Altonorte copper smelter [1], in which the feed matte is nearly white metal, which implies shorter converter cycles (roughly 3.5 hours), and exponentially more schedules to consider. Nonetheless, both of these smelters share the restriction that there can be no more than two simultaneous blowing operations, hence $\overline{n}_{\text{Crit}} = 2$.

Indeed, the three-component CO decomposition (Figure 5.3) is especially descriptive of converting aisles which are limited by the number of simultaneous blowing operations. This is often related to environmental legislation, stating that nearly all of the SO_2 must be captured and converted into acid. For the Chuquicamata and Altonorte smelters in Chile, this causes a restriction of the form $\overline{n}_{\text{Crit}} = 2$. The restriction is even more severe for the Rönnskär smelter in Sweden [75, 76], such that $\overline{n}_{\text{Crit}} = 1$.

Under certain circumstances, it can happen that there is a shortage of ancillary objects from the very beginning of the cycle, so that the precritical stage is omitted (Figure 5.5a); such is the case when there is excessive converting capacity in relation to the smelting, or when there a lack of charging cranes or charging ladles. If the shortage of ancillary objects extends to the end of cycle, then the postcritical stage is omitted (Figure 5.5b); such is the case when there is a shortage of downstream capacity (e.g. a lack of fire refining furnaces), or a lack of discharging cranes or product ladles. The omission of the pre- or the post-critical stage is nonetheless supported by the general CO decomposition (Figure 5.3).

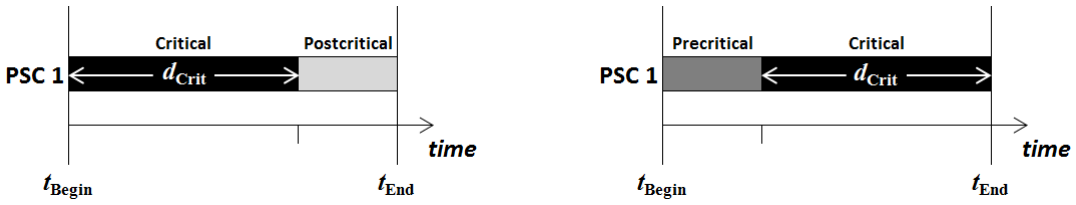


Figure 5.5: Shortage of ancillary objects at (a) the beginning of cycle and (b) the end of the cycle

In order to apply the CO decomposition, several adaptations must be made to the MILP formulation, which are in addition to those described in Subsection 5.1.1. Firstly the set of converter transition types $\mathcal{T}_{\text{PSC}}$ should consist of four disjoint subsets,

$$\mathcal{T}_{\text{PSC}} = \{\text{EndPreviousCycle}\} \bigcup \mathcal{T}^{-}_{\text{PSC,Crit}} \bigcup \mathcal{T}_{\text{PSC,Crit}} \bigcup \mathcal{T}^{+}_{\text{PSC,Crit}} \tag{5.8}$$

The sets $\mathcal{T}^{-}_{\text{PSC,Crit}}$, $\mathcal{T}_{\text{PSC,Crit}}$ and $\mathcal{T}^{+}_{\text{PSC,Crit}}$ are the transition types that constitute precritical, critical and post-critical stages, respectively.

Equation 5.2 causes EndPreviousCycle to precede the other transition types, $\mathcal{T}^{-}_{\text{PSC,Crit}} \bigcup \mathcal{T}_{\text{PSC,Crit}} \bigcup \mathcal{T}^{+}_{\text{PSC,Crit}}$, which form the current cycle. A similar topological ordering must be extended throughout the three stages, such that

$$\left(\mathcal{T}_{\text{PSC,Crit}} \bigcup \mathcal{T}^{+}_{\text{PSC,Crit}}\right) \bigcap \mathcal{T}^{-}_{\text{PSC}k} = \varnothing \tag{5.9}$$

for all $k \in \mathcal{T}^{-}_{\text{PSC,Crit}}$, and

$$\mathcal{T}^{+}_{\text{PSC,Crit}} \bigcap \mathcal{T}^{-}_{\text{PSC}k} = \varnothing \tag{5.10}$$

for all $k \in \mathcal{T}^{-}_{\text{PSC,Crit}} \bigcup \mathcal{T}_{\text{PSC,Crit}}$. Equation 5.9 does not allow the converter to regress into the precritical stage, once it has passed into critical and postcritical stages. Similarly, Equation 5.10 does not allow the converter to regress into the precritical and critical stages, once it has passed into the postcritical stage.

If the cycle does not extend into the critical stage, then there is no scarcity of ancillary objects, and the CO decomposition is meaningless. Thus it can be presumed that

$$\text{EndCurrentCycle} \in \mathcal{T}_{\text{PSC,Crit}} \bigcup \mathcal{T}^{+}_{\text{PSC,Crit}} \tag{5.11}$$

The typical scenario of Chuquicamata, Altonorte, Rönnskär, etc. is described by EndCurrentCycle $\in \mathcal{T}^{+}_{\text{PSC,Crit}}$. However the condition depicted in Figure 5.5b is described by EndCurrentCycle $\in \mathcal{T}_{\text{PSC,Crit}}$, for which Equation 5.10 causes $\mathcal{T}^{+}_{\text{PSC,Crit}} = \varnothing$. Considering the combination of Equations 5.2 and 5.10-11, it can be reasoned that $\mathcal{T}^{+}_{\text{PSC,C}} = \varnothing$ if and only if EndCurrentCycle $\in \mathcal{T}_{\text{PSC,Crit}}$.

Unlike the full MILP formulation, the CO decomposition does not fully describe the sharing of ancillary objects within the converting aisle. Every object that is occupied during *part* of the critical stage is treated as if it were occupied during *all* of the critical stage, neglecting the possibility of early or temporary release of these objects. This restriction is erroneous, but is acceptable for assigning bulky equipment, such as the offgas handling

system, which does not undergo frequent reassignments.

5.1.3 Dominance Condition for the Critical Stage

The Critical Overlap decomposition provides a worthwhile link between local optimality for a single converting cycle, and global optimality for an entire converting schedule. The efficient use of ancillary resources can be measured by the production in one cycle, divided by the duration of the critical stage. This ratio is of particular importance when the critical stage dominates the cycle.

To quantify this dominance condition, the duration of the critical stage d_{Crit} is compared to the total duration of the cycle d_{Cycle}. Figure 5.6 considers $(n_{\text{PSC}}, \overline{n}_{\text{Crit}}) = (2, 1)$; as long as $d_{\text{Crit}} \geq \frac{1}{2} d_{\text{Cycle}}$, it is possible to maintain maximum utility, $n_{\text{Crit}} = \overline{n}_{\text{Crit}}$, indefinitely throughout the schedule. Similarly, Figure 5.7 considers $(n_{\text{PSC}}, \overline{n}_{\text{Crit}}) = (3, 2)$, and it is possible to maintain $n_{\text{Crit}} = \overline{n}_{\text{Crit}}$ as long as $d_{\text{Crit}} \geq \frac{2}{3} d_{\text{Cycle}}$.

More generally, when considering one kind of cycle,

$$d_{\text{Crit}} \geq \left(\frac{\overline{n}_{\text{Crit}}}{n_{\text{PSC}}} \right) d_{\text{Cycle}} \tag{5.12}$$

is a sufficient condition to ensure that there are feasible schedules that satisfy $n_{\text{Crit}} = \overline{n}_{\text{Crit}}$ at all times, and are hence optimal. This condition is apparent since, in order that $\overline{n}_{\text{Crit}}$ out of the n_{PSC} converters should always be in the critical stage, the average converter should be in the critical stage at least $(\overline{n}_{\text{Crit}}/n_{\text{PSC}})$ of the schedule duration.

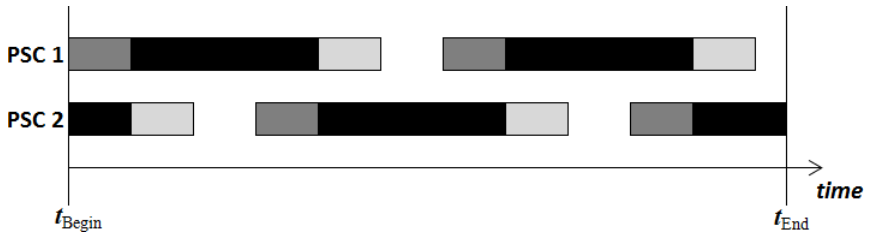
Cycles which satisfy condition 5.12 are termed critically dominant, meaning that d_{Crit} dominates d_{Cycle}. This is typical of the Rönnskär and Chuquicamata smelters which are described by Figure 5.6a and Figure 5.7a, respectively. The cycles at the Altonorte smelter are better described by Figure 5.7c, hence they are not critically dominant; again, this is related to the high grade of the furnace matte.

To incorporate the critical dominance condition into the MILP, the total cycle duration $d_{\text{Cycle}} \in \mathbb{R}^+_\circ$ is computed as

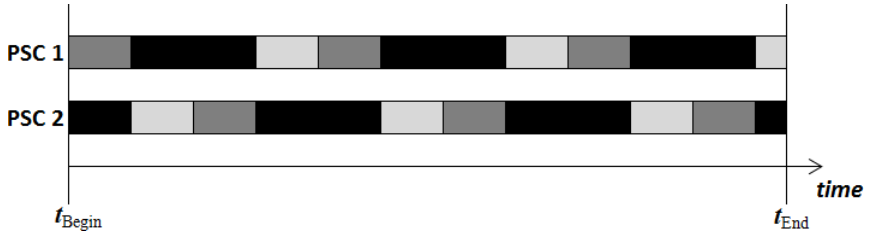
$$d_{\text{Cycle}} = \sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ} d^l \tag{5.13}$$

Any transitions which are undetermined in the current cycle will be such that $d^l = 0$, in accordance with Equation 4.4.

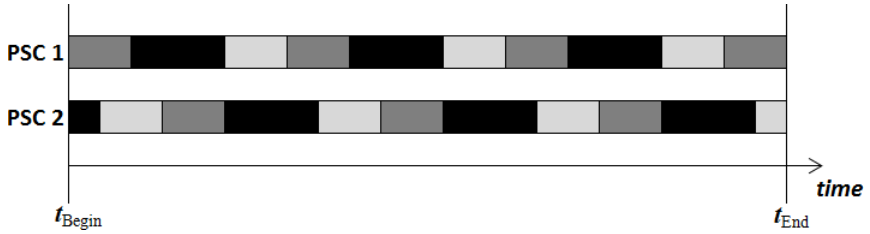
The critical duration $d_{\text{Crit}} \in \mathbb{R}^+_\circ$ is implemented in conjunction with the nonnegative



(a) $d_{\text{Crit}} > \frac{1}{2} d_{\text{Cycle}}$



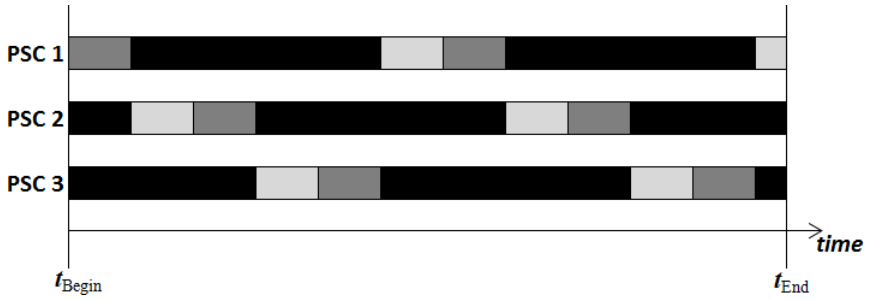
(b) $d_{\text{Crit}} = \frac{1}{2} d_{\text{Cycle}}$



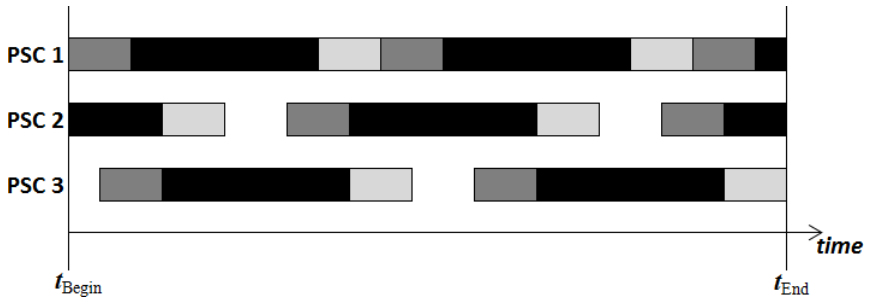
(c) $d_{\text{Crit}} < \frac{1}{2} d_{\text{Cycle}}$

Figure 5.6: Optimal production schedules for two-converter systems, having different d_{Crit} to d_{Cycle} ratios

(a) $d_{\text{Crit}} > \frac{2}{3} d_{\text{Cycle}}$



(b) $d_{\text{Crit}} = \frac{2}{3} d_{\text{Cycle}}$



(c) $d_{\text{Crit}} < \frac{2}{3} d_{\text{Cycle}}$

Figure 5.7: Optimal production schedules for a three-converter system, having different d_{Crit} to d_{Cycle} ratios

variables $d^l_{\text{Crit}} \in \mathbb{R}^+_\circ$, such that

$$d^l_{\text{Crit}} = \begin{cases} d^l & \text{Type}^l \in \mathcal{T}_{\text{PSC,Crit}} \\ 0 & \text{otherwise} \end{cases}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. Thus d^l_{Crit} is the contribution to d_{Crit} which is due to transition l,

$$d_{\text{Crit}} = \sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ} d^l_{\text{Crit}} \tag{5.14}$$

The contributions are determined according to the following inequalities,

$$\begin{aligned} d^l_{\text{Crit}} &\geq d^l - \overline{d}\left(1 - \sum_{k \in \mathcal{T}_{\text{PSC,Crit}}} \beta^l_{\text{Type}k}\right) && (5.15) \\ &\leq \overline{d} \sum_{k \in \mathcal{T}_{\text{PSC,Crit}}} \beta^l_{\text{Type}k} && (5.16) \\ &\leq d^l && (5.17) \end{aligned}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. If l is part of the critical stage, then $\beta^l_{\text{Type}k} = 1$ for some $k \in \mathcal{T}_{\text{PSC,Crit}}$, and Equations 5.13 and 5.15 cause $d^l_{\text{Crit}} = d^l$, while Equation 5.14 is slack; otherwise if l is not part of the critical stage then Equation 5.14 acts with the nonnegativity condition so that $d^l_{\text{Crit}} = 0$, as Equations 5.13 and 5.15 are slack.

By introducing Equations 5.12-17 into the MILP, the optimization is restricted to critically dominant cycles. However, a critically dominant cycle may not always be feasible, as demonstrated in Section 5.4.

5.1.4 Maximizing the Productivity of a Single Converting Cycle

The design of a cycle is based primarily on the maximization of (Production/d_{Crit}) ratios. These ratios provide a direct link between the optimality of a critically dominant cycle, and the optimality of the larger schedule. But even for cycles that are not critically dominant, these ratios are valid heuristics that analyze the utilization of ancillary objects.

Considering the global objective described by Equation 4.98, the relevant local objective is

$$\max f = \frac{\sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ} \left(\sum_{j \in \mathcal{F}_{\text{NG}}} \chi_{v,j} v^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} \chi_{\beta,k} \beta^l_{\text{Type}k} \right)}{d_{\text{Crit}}} \tag{5.18}$$

which is fractional, hence not directly supported by the MILP formulation. The Charnes-Cooper transformation can produce an equivalent objective function that is linear, however the branching is complicated in a way that is not supported by standard linear programming solvers (Appendix B.6).

Rather than working directly with the fractional objective (Equation 5.18), a sequence of iterations $i \in \{1, 2, \ldots\}$ can be applied, in which the critical duration is bounded above by $\overline{d}_{\text{Crit},i}$; this upper bound is varied from iteration to iteration, and satisfies $0 < \overline{d}_{\text{Crit},i} < (t_{\text{End}} - t_{\text{Begin}})$.

Each iteration is composed of two MILP computations. The first is to maximize production, hence the objective function,

$$\max f_1 = \sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ} \left(\sum_{j \in \mathcal{F}_{\text{NG}}} \chi_{v,j} v^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} \chi_{\beta,k} \beta^l_{\text{Type}k} \right) \tag{5.19}$$

with the additional restriction,

$$d_{\text{Crit}} \leq \overline{d}_{\text{Crit},i} \tag{5.20}$$

The resulting optimal value f^*_{1i} is then used in the second computation, which is to minimize the critical duration, while maintaining the optimal production; thus the second objective is to minimize

$$\min f_2 = d_{\text{Crit}} \tag{5.21}$$

given the additional restriction

$$\sum_{l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ} \left(\sum_{j \in \mathcal{F}_{\text{NG}}} \chi_{v,j} v^{jl} + \sum_{k \in \mathcal{T}_{\text{PSC}}} \chi_{\beta,k} \beta^l_{\text{Type}k} \right) \geq f^*_{1i} - \epsilon_{f1} \tag{5.22}$$

where $\epsilon_{f1} \approx 0$ is used to promote numerical stability; $\epsilon_{f1} = 0.0001 f^*_{1i}$ has been found to work well in practice. It is inconsequential whether or not equation 5.20 is included in the second computation because it is already implied by the construction of Equations 5.21-22. The resulting optimal value is denoted f^*_{2i}.

Each iteration generates a candidate for the optimization of optimal value of Equation 5.18, or rather

$$f^* \approx \max_{i \in \{1,2,\ldots\}} \left(\frac{f^*_{1i}}{f^*_{2i}} \right) \tag{5.23}$$

A complete sweeping of all possible $\overline{d}_{\text{Crit},i}$ values will necessarily yield the optimal solution for

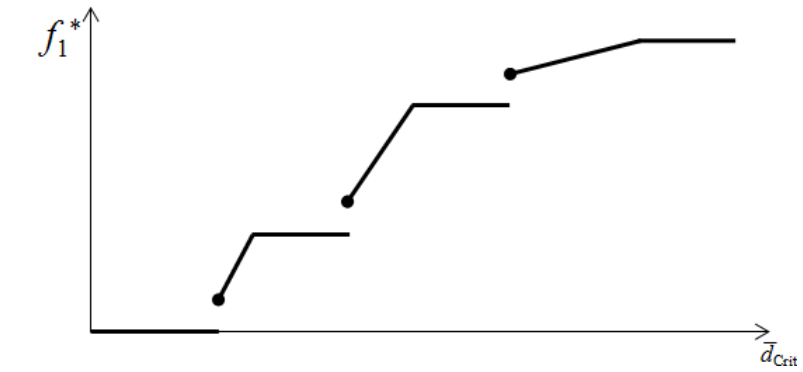
Equation 5.18, as will be argued below. In practice, it is sufficient to use equi-spaced values of $\overline{d}_{\text{Crit},i}$ at quarter hour intervals, i.e. $\overline{d}_{\text{Crit},i} = \frac{i}{4}$ for $i = 1$ to $\lfloor 4(t_{\text{End}} - t_{\text{Begin}}) \rfloor$. Additional iterations can then be performed to give a more refined search. With enough iterations, detailed plots can be obtained for f^*_{1i}, f^*_{2i} and $\left(\frac{f^*_{1i}}{f^*_{2i}}\right)$, as depicted in Figure 5.8.

Figure 5.8a illustrates that f^*_{1i} is a nondecreasing function of $\overline{d}_{\text{Crit},i}$, as a larger value of $\overline{d}_{\text{Crit},i}$ implies a larger feasibility region in the first computation. There may be intervals of $\overline{d}_{\text{Crit},i}$ values for which Equation 5.20 is continually active ($d_{\text{Crit}} = \overline{d}_{\text{Crit},i}$), which correspond to the strictly increasing segments in Figure 5.8a. These segments are followed by flat regions, in which Equation 5.20 is dominated by other constraints. Certain of the flat regions can be followed by discrete jumps; this occurs at $\overline{d}_{\text{Crit}}$ values which allow either a numerical change in the delivery units u^j, or a categorical change in the transition sequence.

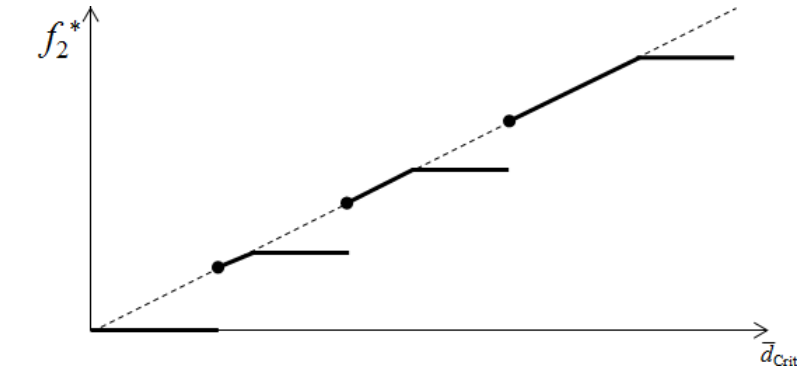
As the production f^*_{1i} increases, the critical duration cannot be shortened, which makes f^*_{2i} also a nondecreasing function of $\overline{d}_{\text{Crit},i}$ (Figure 5.8b). The increasing segments in Figure 5.8b coincide with those of Figure 5.8a, as do the flat regions and the discrete jumps. In particular, the increasing segments are characterized by $f^*_{2i} = \overline{d}_{\text{Crit},i}$, because this duration is necessary in order to maintain the production level prescribed by Equation 5.22.

The ratio of the linear segments from Figures 5.8a and 5.8b form the curved segments in Figure 5.8c, which can be either increasing or decreasing. Furthermore, the maximum of $\left(\frac{f^*_{1i}}{f^*_{2i}}\right)$ can always be observed at the one of the endpoints of the curved segments, where the ratio ceases to increase.

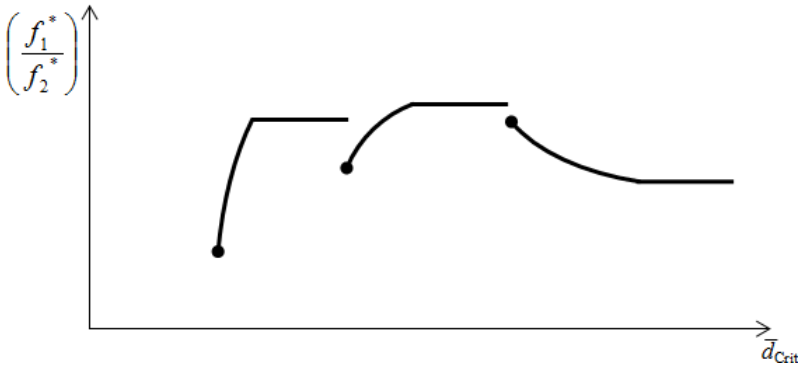
The SC-PSC formulation, together with the notion of critical dominance, presents a methodology for designing and evaluating converting cycles. The managers of a smelter can use these concepts to simulate the impact of a proposed upgrade, on a cycle-to-cycle basis, hence to justify or deny such an upgrade.



(a)



(b)



(c)

Figure 5.8: Construction of the productivity ratio objective for the SC-PSC Problem

5.2 Software Systems

5.2.1 AMPL and CPLEX

AMPL ("A Mathematical Programming Language") is a computer programming language whose syntax is remarkably similar to that of mathematical programming [77]. AMPL is especially well-suited to implement the constraints defined in Chapter 4, as well as the sets and parameters defined in Chapters 2 and 3. Once an optimization problem is programmed into AMPL, the platform can then solve the problem by appealing to a variety of external solvers; the most well-established MILP solver is CPLEX [71, 72].

AMPL computations rely on three types of source files, identified by the following extensions types:

- *mod* files declare all of the underlying sets and parameters which form the model. They also contain the constraints and objective functions that link together the underlying sets and parameters.
- *dat* files contain the specifications of the particular instance of the problem, including the list of members that form each of the underlying sets and the numerical parameter values.
- *run* files contain scripted instructions of when to load the data and the models, execute the solver and organize the results.

Appendix C contains AMPL files that were developed for the current research. The *dat* files vary depending on the parameters. However the *mod* and *run* files do not.

The SC-PSC formulation alternates between two objectives (Equations 5.19 and 5.21), which is managed through the *run* file (Appendix C.3). The *run* file also activates and deactivates Equation 5.22, which is relevant under the second objective only.

To optimize the ratio objective (Equation 5.18), the *run* file calls on the CPLEX solver to perform a sequence of optimizations, resulting in graphs similar to those of Figure 5.8. First there is a sweep of $\overline{d}_{\mathrm{Crit},i}$ values from 0 to 12 hours, using quarter hour intervals. Then there is a more refined sweep, using 5 minute intervals, exploring the half hour that surrounds the best result from the first sweep. Considering both sweeps, there is a total of 52 iterations. The best solution from the 52 iterations is saved into memory.

The dependency constraints (Equations 4.8-4.14) are deactivated throughout the 52 iterations. They are reactivated afterward, and CPLEX is then used to construct a feasible

schedule for an ancillary object, given the fixed PS schedule that had been saved into memory. This is not necessarily the most practical way to coordinate the ancillary objects, but it effectively verifies the formulation of Subsection 4.1.2, which is otherwise irrelevant to the SC-PSC Problem.

The results are stored in an output text file, including the optimal solution, the ancillary assignment, the information required to produce the aforementioned graphs (Figure 5.8), and the computation times. This output file follows the *csv* (comma-separated-values) format, which is easily imported into Microsoft Excel© and other common software packages [78].

CPLEX is only one of the many commercial solvers that is supported by AMPL. An alternative is the MOSEK solver [79], which supports quadratic constraints, and would hence allow the direct implementation of concentration and temperature variables (Section 6.1). The MINLPBB solver [80] works for mixed integer nonlinear programs, and can thus accept a linear-fractional objective (e.g. Equation 5.18). In addition to CPLEX, MOSEK, MINLPBB, and the other commercially available solvers, AMPL allows users to incorporate their own solvers [72].

Future work may include testing and developing solvers to support variants of the Peirce-Smith Converter Problem which are more demanding than the SC-PSC. Even so, the AMPL source code will remain largely unchanged, because the platform is designed to accommodate these various solvers.

5.2.2 Excel and VBA

Using the commonly available Microsoft Excel©, three macro-enabled workbooks [81] have been constructed to facilitate the preprocessing of problem data and postprocessing of computational results for the SC-PSC Problem.

The first two workbooks describe a copper production (Section 5.3), firstly through the Simplified Copper PS formulation, and secondly through the General Nickel-Copper formulation. The third workbook applies the General Nickel-Copper formulation to describe a case of nickel-copper converting (Section 5.4). Interestingly, the workbooks all rely on the same *mod* file (Appendix C.1) and run file (Appendix C.3), since the underlying mathematical structures are equally valid for both the general and the simplified formulation.

Each workbook is divided into a sequence of sheets having the following labels,

- Sheet 1: Start

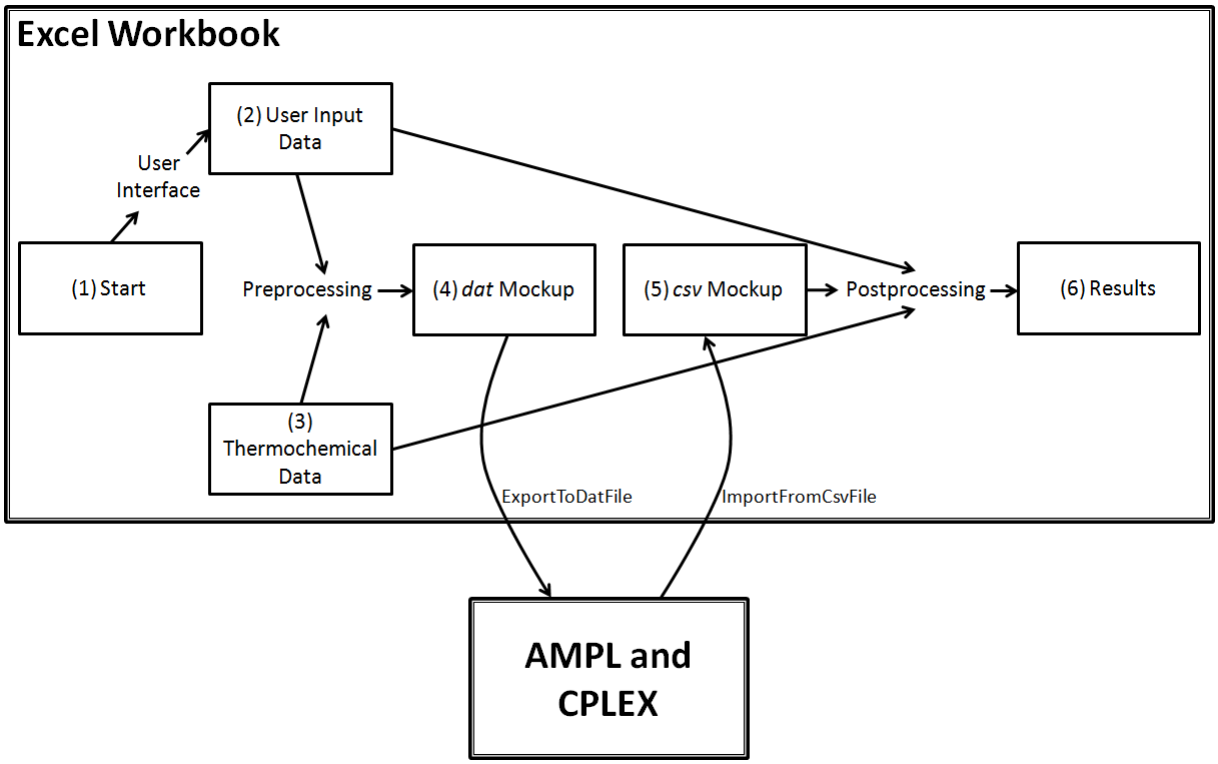


Figure 5.9: Interaction between Excel and the optimization platform which consists of AMPL and CPLEX

- Sheet 2: User Input Data
- Sheet 3: Thermochemical Data
- Sheet 4: *dat* Mockup
- Sheet 5: *csv* Mockup
- Sheet 6: Results

Figure 5.9 illustrates the dependencies between these sheets, and their interaction with the optimization platform.

Sheet 1 is the starting point for the user interface, which was programmed using Excel's Visual Basic for Applications (VBA) module [82]. It presents a series of forms so that the user may enter problem data, which is then stored in Sheet 2.

All of the pre- and post-processing of data is automated within the workbooks. For instance, the thermal inputs are based on temperature, whereas the thermal constraints (Subsection 4.5.4) are based on heat; the workbooks automatically convert the user-input

temperatures (Sheet 2) into heats, using the thermochemical data (Sheet 3) in conjunction with the formulas of Chapter 3. The workbooks also include formulas that compute the various upper and lower bounds presented in Chapter 4. Although it is possible to program these formulas into AMPL instead of Excel, this approach would be more cryptic, and more tedious to verify.

Sheet 4 contains all of the text and numbers used to create a *dat* file. The numerical values are automatically computed, depending on the content of Sheets 2 and 3. The ExportToDatFile macro has been programmed within the workbook to automatically export the problem data from Sheet 4 into the *dat* file; the macro simply copies the content of Sheet 4 into a blank file, line by line, to create the *dat* file.

The resulting *dat* file is then read into the AMPL platform, as directed by the *run* file. AMPL then calls on the CPLEX solver to perform the sequence of optimizations, and finally organizes the results into a *csv* file, as described in Subsection 5.2.1.

The content of the *csv* file is then copied into Sheet 5. This is performed using the ImportFromCsvFile macro that has been programmed into the workbook. Subsequently, Sheet 6 draws upon the content of Sheet 5 to tabulate the results. Sheet 6 also depends on Sheets 2 and 3 in order to relate the results to the input, and to apply the thermochemical formulas of Chapter 3.

The intermediate computations (Section 4.3) and forward computations (Section 4.4) have all been programmed into Excel. Therefore, *csv* does not include the state and the intermediate variables; only the transition variables are transmitted. By comparing the Excel results to the AMPL results, it is verified that the intermediate and forward computations are properly incorporated into the MILP.

The raw optimization results of Sheet 5 include heat values which are transformed into temperatures using the specially designed ComputeTemperature function. Following the approach of Subsection 3.2.4, the function has six arguments $(w_{\mathrm{H}}, w_{\mathrm{H}}^{\mathrm{Ref}}, A, B, C, D)$. If the function fails to locate an appropriate temperature after 50 iterations, then it returns an error. Otherwise, it returns a temperature value, accurate to eight significant digits.

Excel provides a user-friendly method to prepare the data and tabulate the results of the SC-PSC Problem. It can eventually be a platform for customized software that will be marketed to consulting companies and smelters.

5.3 Sample Computations

5.3.1 Sample Computations for a Copper PS Converter

The MILP model includes parameters and equations that can accommodate virtually any Peirce-Smith system. However, a typical user of the system would work with a reduced set of parameters that are accessible through the user interface (Figure 5.10), and are based on the particular instance of the problem. The input data described in Tables 5.1-5.4 are based on the Rönnskär smelter [75, 76], but could easily be adapted to other contexts.

The dynamics are described by Figure 5.2a, and the objective is to maximize the rate of feed matte conversion, given a restricted access to the offgas handling. In this case, the critical period begins with the first blowing action, and ends with the final blowing action. In the first section of Table 5.1, the scheduling horizon is an expression of t_{End}; the duration of the converter cycle may not exceed 12 h. The critical dominance condition (Equation 5.12) considers the number of converters in operation, $n_{\text{PSC}} = 2$, and the offgas handling capacity, $\overline{n}_{\text{Crit}} = 1$.

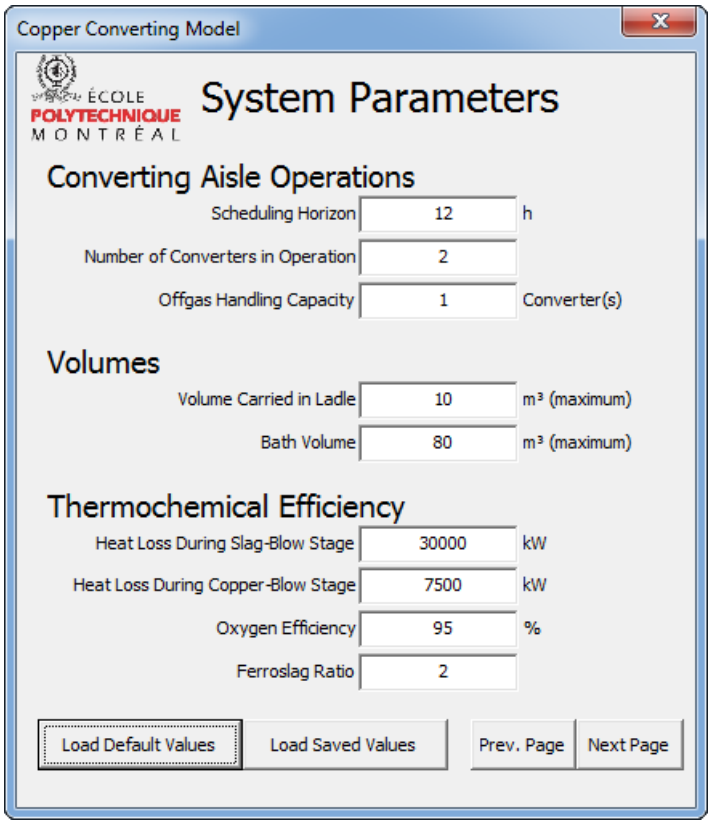


Figure 5.10: First page of the user interface for sample copper PSC computations

Table 5.1: User input for sample copper PSC computations (System Parameters)

Converting Aisle Operations		
Scheduling Horizon:	12	h
Number of Converters in Operation:	2	
Offgas Treatment Capacity:	1	Converter
Volumes		
Volume Carried in Ladle:	10	m^3 (maximum)
Bath Volume:	80	m^3 (maximum)
Thermochemical Efficiency		
Heat Loss During Slag-Blow Stage:	30000	kW
Heat Loss During Copper-Blow Stage:	7500	kW
Oxygen Efficiency:	95	%
Ferroslag Ratio:	2	

Table 5.2: User input for sample copper PSC computations (Converting Cycle)

Initial Charge		
Initial Charge Duration:	0.5	h
Initial Matte Delivery:	6	Ladles (maximum)
Slag-Blow Stage		
Duration of a Single Slag-Blow Action:	0.5	h (minimum)
Excess Silica in Slag:	10	% (maximum)
Skim Duration:	0.125	h
Recharge Duration:	0.125	h
Recharges per Cycle:	2	(maximum)
Matte Delivery per Recharge:	3	Ladles (maximum)
Copper-Blow Stage and Final Discharge		
Duration of a Single Copper-Blow Action:	1	h (minimum)
Scrap Charge Duration:	0.125	h
Scrap Charges per Cycle:	2	(maximum)
Copper Oxidation:	5	% of total copper
Final Discharge Duration:	0.5	h

The matte may only be fed as full ladles, either during the initial charge, or during subsequent recharges. The flux and reverts may be fed during the initial charge and the recharges, and may also be fed during the blowing operation. Lastly, the copper scrap may only be added during the Copper-Blow stage, when the blowing is paused. The flux, reverts and scrap are unmodulated, and the feed matte is modulated.

The user interface provides a layer of flexibility regarding the presentation of the parameters. For example, the heat loss parameters are entered in terms of kW (Figure 5.10), while the model functions in MJ/h. The conversion from kW into MJ/h is performed by the Excel spreadsheet, so that the *dat* file contains the appropriate numerical value. For instance, an environmental heat loss of 30000 kW corresponds to 108000 MJ/h (See Appendix C.2, and Subsection 4.3.7).

In Table 5.2, the Initial Charge Duration is presented as a single parameter that has been set to 0.5 h. However, the underlying model may accommodate a more complicated relation (Equation 4.32), which considers to the charging units and volumes. The skimming, recharging, scrap charge and final discharge durations are also fixed in a similar way.

Also in Table 5.2, the entry "Excess Silica in Slag" describes an upper bound on the percentage of unreacted silica SiO_2 that may be present in the slag. Without this upper bound, there would be an excessive (unpractical) use of flux. This condition has been implemented using Equation 4.97.

Table 5.2 also places an upper limit on the number of ladles which can be delivered during a recharge transition, implemented using Equation 4.28. This might be related to the availability of cranes, for example.

Additionally, there is an overblowing condition, stating that 5% of the total copper should be oxidized. As described in Chapter 1, this ensures a higher degree of sulfur elimination, and reduces the workload of the fire refining furnaces. This overblow condition has been implemented in general linear form, using two adaptations of Equation 4.106. Firstly,

$$(1-x)\, w_{\mathrm{Cu,Cu_2O}} \overline{m}_{\mathrm{Cu_2O}} \beta^{l}_{\mathrm{Type,CopperBlow}} - x m^{l}_{\mathrm{CuLiq,Prod}} + (1-x)\, w_{\mathrm{Cu,Cu_2O}} m^{l}_{\mathrm{Cu_2O,Prod}} \leq (1-x)\, w_{\mathrm{Cu,Cu_2O}} \overline{m}_{\mathrm{Cu_2O}}$$

in which $x = 0.05$ is the proportion of copper that is to be oxidized; when transition l is a CopperBlow, the $\beta^{l}_{\mathrm{Type,CopperBlow}}$ term cancels out the righthand side, and prevents more than 5% oxidation. Secondly,

$$x \overline{m}_{\mathrm{CuLiq}} \beta^{l}_{\mathrm{Type,EndCurrentCycle}} + x m^{l-}_{\mathrm{CuLiq,RetProd}} - (1-x)\, w_{\mathrm{Cu,Cu_2O}} m^{l-}_{\mathrm{Cu_2O,RetProd}} \leq x \overline{m}_{\mathrm{CuLiq}}$$

Table 5.3: User input for sample copper PSC computations (Feeds)

Matte	Cu:	60	wt%
Flux	SiO_2:	85	wt%
	CaO:	5	wt%
	Al_2O_3:	5	wt%
	MgO:	5	wt%
Blast			
	Blast Flowrate in Slag-Blow:	45000	Nm^3/h
	Enrichment in Slag-Blow:	25	vol%O2
	Blast Flowrate in Copper-Blow:	45000	Nm^3/h
	Enrichment in Copper-Blow:	25	vol%O2
Reverts	Fe:	24.04	wt%
	Cu:	50.14	wt%
	S:	6.18	wt%
	Si:	5.99	wt%
	Ca:	0	wt%
	Al:	0	wt%
	Mg:	0	wt%
	O:	13.65	wt%
	Density:	5.5272	T/m^3
	Specific Heat of Formation:	-172.76	MJ/T
	Heat Capacity Coefficient, A:	0.46902	J/(g∘C)
	Heat Capacity Coefficient, B:	0.00044730	$J/(g(°C)^2)$
	Heat Capacity Coefficient, C:	-7890.9	(J°C)/g
	Heat Capacity Coefficient, D:	0	$J/(g(°C)^3)$

When l is to end the current cycle, the righthand side is canceled, so that a minimum of 5% oxidation is required to proceed with the transition. The combined effect of both inequalities is that there must be exactly 5% oxidation as the Copper-Blow is ending, and that there cannot be any more than 5% oxidation during the Copper-Blow.

Table 5.3 describes a typical matte grade, and flux composition. The blast rates and oxygen enrichments are based on the Rönnskär smelter [75]. The reverts data is taken from Table 3.4, and is admittedly only an estimation. More advanced studies would require an analysis of industrial reverts, which is likely to differ substantially from smelter to smelter.

Table 5.4: User input for sample copper PSC computations (Temperatures)

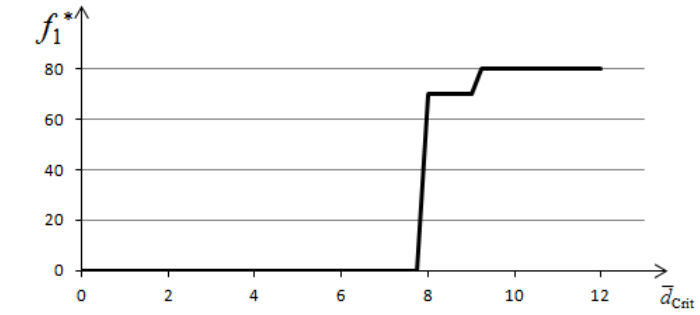
Feed Temperatures		
Matte Feed Temperature:	1200	°C
Cold Feed Temperature:	30	°C
Blowing Temperatures		
Blast Temperature:	50	°C
Bath Temperature During Slag Blow:	1050	°C (minimum)
Bath Temperature During Slag Blow:	1250	°C (maximum)
Bath Temperature During Copper Blow:	1150	°C (minimum)
Bath Temperature During Copper Blow:	1250	°C (maximum)
Product Temperatures		
Slag Temperature:	1230	°C
Offgas Temperature During Slag Blow:	1200	°C
Offgas Temperature During Copper Blow:	1200	°C
Final Discharge Temperature:	1150	°C (minimum)
Final Discharge Temperature:	1250	°C (maximum)

Table 5.4 demonstrates the shortcomings of the MILP formulation, with regard to temperature data. While it is appropriate for the input temperatures to be fixed (i.e. the feed and blast temperatures), the remaining temperatures should all be allowed to vary within ranges in response to operational decisions. The MILP formulation could not support ranges for the slag, nor for the offgas, as discussed in Subsections 4.5.4 and Section 6.1.

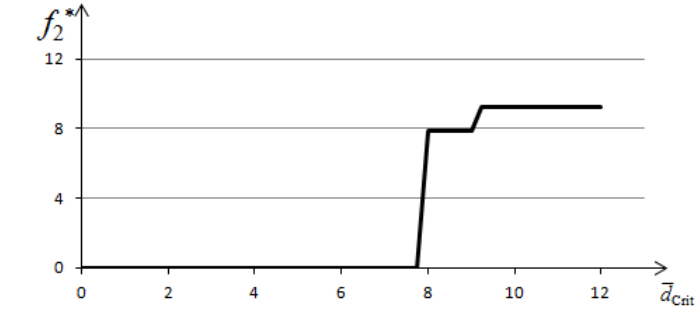
The data of Tables 5.1-4 has been entered into the General Nickel-Copper Formulation, as well as the Simplified Copper Formulation. Both computations obtain the same objective value, 8.867 m^3 feed matte / critical hour, but with different solutions, as described below. The general formulation requires 114.5 seconds of computation time, while the simplified formulation requires 103.5 seconds.

Figure 5.11 depicts the objective functions that were obtained throughout the 52 iterations. As expected, identical objective values are observed for both the General Nickel-Copper Formulation, and the Simplified Copper Formulation. In both cases, the optimal ratio objective is first observed at $\overline{d}_{\text{Crit}} = 8$ h, which is the 32^{nd} iteration.

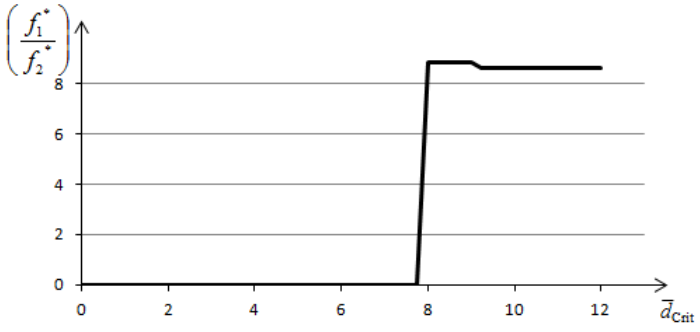
The two solutions may be compared by examining the resulting Gantt charts (Figure 5.12), and the feed schedule (Table 5.5). Both schedules employ a critical duration of 7.89 h to process 7 ladles of matte. However, the duration of the individual blowing actions are



(a)

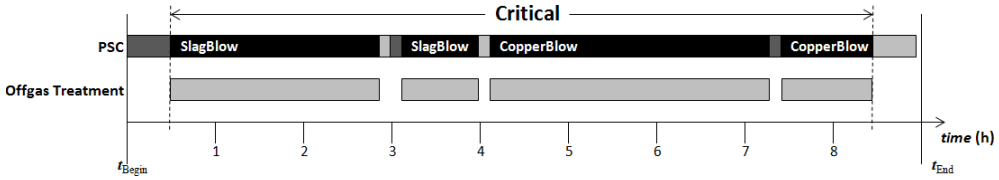


(b)

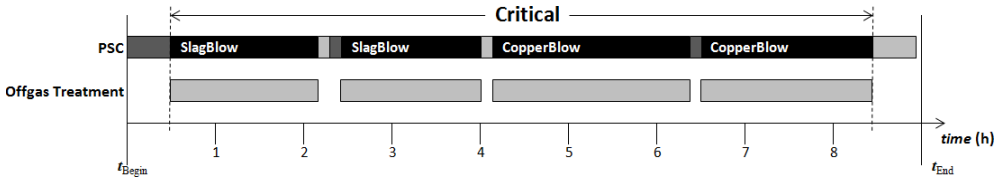


(c)

Figure 5.11: Objective functions from sample copper PSC computations



(a) General Nickel-Copper Formulation

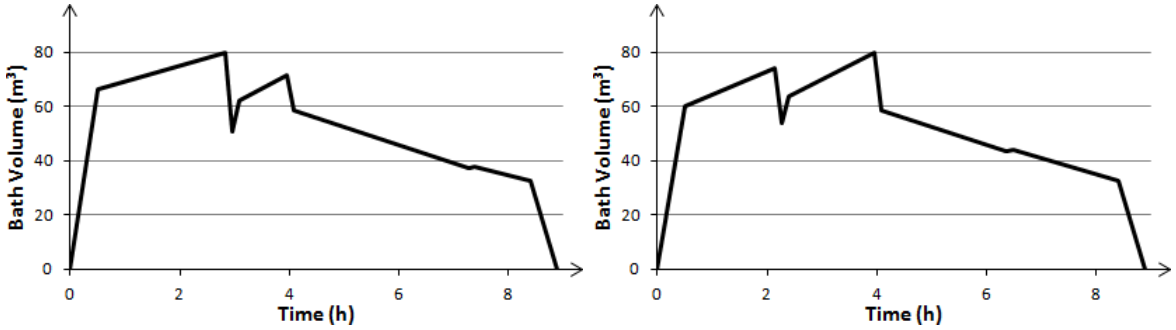


(b) Simplified Copper Formulation

Figure 5.12: Optimal Gantt charts from sample copper PSC computations

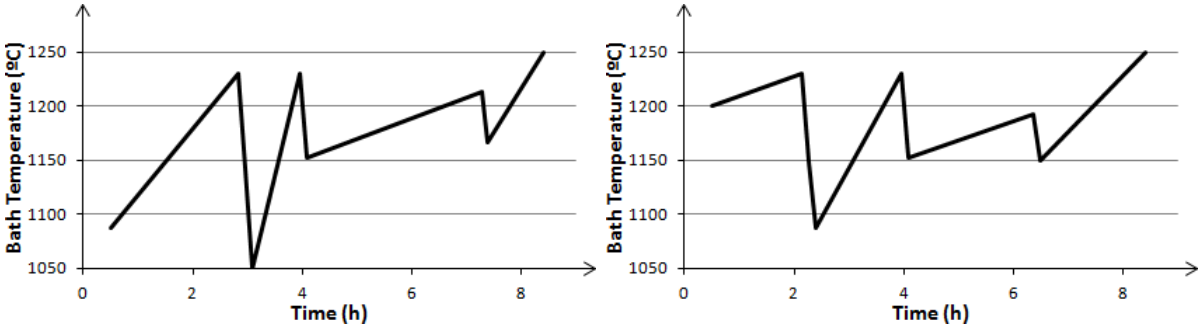
Table 5.5: Feed tonnages from copper PSC computations

	General Formulation					**Simplified Formulation**				
Transition	1	2	4	5	8	1	2	4	5	8
Start Time (h)	0	0.5	2.96	3.09	7.27	0	0.5	2.26	2.389	6.36
Finish Time (h)	0.5	2.84	3.09	3.97	7.39	0.5	2.14	2.39	3.97	6.49
Charging Feed										
Feed Matte	316.7	0	52.8	0	0	316.7	0	52.8	0	0
Flux	16.8	0	0	0	0	0	0	0	0	0
Reverts	0	0	6.4	0	0	0	0	0	0	0
Copper Scrap	0	0	0	0	5.6	0	0	0	0	5.6
Blowing Feed										
Flux	0	0	0	6.1	0	0	11.7	0	11.1	0
Reverts	0	28.7	0	15.2	0	0	20.8	0	29.5	0



(a) General Nickel-Copper Formulation (b) Simplified Copper Formulation

Figure 5.13: Bath volume from sample copper PSC computations



(a) General Nickel-Copper Formulation (b) Simplified Copper Formulation

Figure 5.14: Bath temperature from sample copper PSC computations

different. For instance, the general formulation obtains the maximum volume content of 80 m^3 at the end of the first Slag-Blow action (Figure 5.13a), whereas the the simplified formulation obtains it at the end of the second Slag-Blow action (Figure 5.13b).

In Figure 5.14 it can be observed that the Slag-Blow actions only attain a maximum temperature value of 1230°C, even though the maximum allowable temperature is 1250°C. This is due to the constraint that skimming must occur at 1230°C, to ensure that the outgoing slag has this temperature. It would be more realistic if the slag temperature were allowed to vary, as described in Section 6.1. Nonetheless, Figure 5.14 demonstrates the proper functioning of the temperature constraints (Subsection 4.5.4), as well as the implementation of Newton's Method, as per Equation 3.26.

The computations presented by Tables 5.1-5 and Figures 5.10-14 demonstrate the use of the MILP model for preliminary copper PSC computations. There are numerous features, such as variable charge time, idle time, etc. which may be more useful in comprehensive smelter studies that would provide appropriate data.

5.3.2 Sample Computations for a Nickel-Copper PS Converter

The General Nickel-Copper Formulation is equally applicable to copper converting problems, as it is to nickel-copper converting problems. The input data is of Tables 5.7-9 is loosely based on the Falconbridge Slag Make Furnace [46], which is relatively large, and is noted for its use of ALSI technology [45].

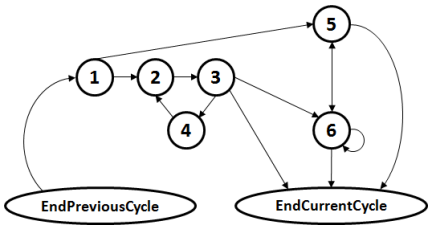


Figure 5.15: Converter transition diagrams for limited access to the smelting furnace. The transition types are numbered (1) InitialCharge, (2) SlagBlow, (3) Skim, (4) Recharge, (5) SlagBlowAndSkimWithoutAnyMoreFeedMatte, (6) ExtendProductionCycleWithoutAnyMoreFeedMatte.

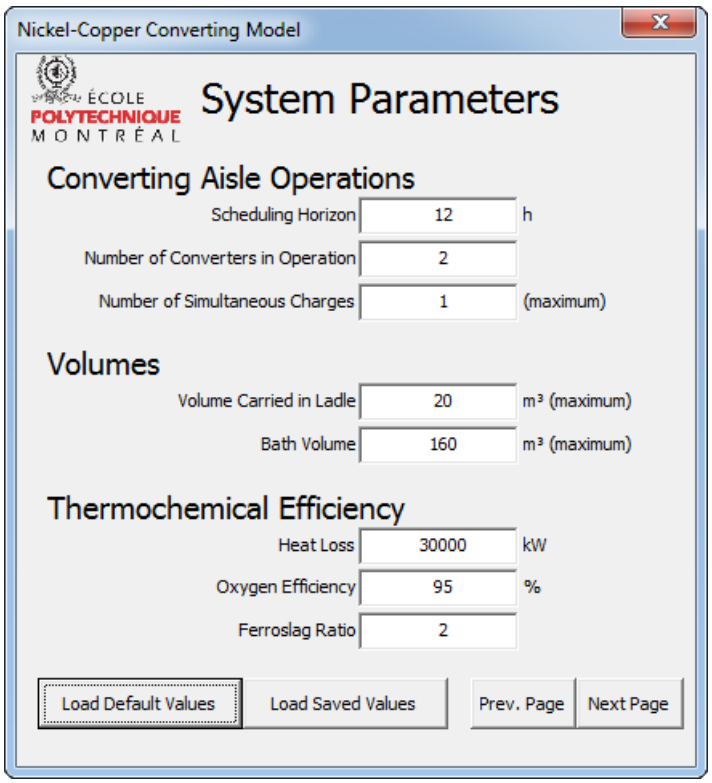


Figure 5.16: First page of the user interface for sample nickel-copper PSC computations

In the current example, the objective is to incorporate as much ferronickel as possible into the cycle, given the limited availability of the smelting furnaces. The ferronickel is regarded as an unmodulated charging feed, which cannot be introduced during the blowing action. The system dynamics are described by Figure 5.15, in which the critical duration includes transitions 1 through 4; when the cycle progresses into transitions 5 or 6, the furnace is free to begin charging another converter. In Figure 5.16 and Table 5.6, “Number of Simultaneous

Table 5.6: User input for sample nickel-copper PSC computations (System Parameters)

Converting Aisle Operations		
Scheduling Horizon:	12	h
Number of Converters in Operation:	2	
Number of Simultaneous Charges:	1	(maximum)
Volumes		
Volume Carried in Ladle:	20	m^3 (maximum)
Bath Volume:	160	m^3 (maximum)
Thermochemical Efficiency		
Heat Loss:	30000	kW
Oxygen Efficiency:	95	%
Ferroslag Ratio:	2	

Table 5.7: User input for sample nickel-copper PSC computations (Converting Cycle)

Charge and Recharge		
Initial Charge Duration:	0.5	h
Initial Matte Delivery:	4	Ladles (maximum)
Initial Ferronickel Delivery:	40	T (maximum)
Recharge Duration:	0.125	h
Matte Delivery during Recharge:	2	Ladles (maximum)
Total Feed Matte:	6	Ladles (minimum)
Blow and Skim		
Duration of a Blow Action:	0.5	h (minimum)
Excess Silica in Slag:	10	% (maximum)
Skim Duration:	0.125	h
Final Discharge		
Final Discharge Duration:	0.5	h

Charges" thus describes the number of converters which may be simultaneously in the critical stage.

As described in Chapter 1, the Doniambo and Sorowako smelters routinely employ PS converting to convert ferronickel into a nickel-sulfide matte, and ultimately to obtain metallic nickel [2]. Otherwise, ferronickel is mainly used as an alloying component of stainless steel [22]. Given the versatility of Peirce-Smith converting, a nickel-copper smelter could conceivably arbitrate between the price of metallic nickel and that of stainless steel. The sample computations describe a mode of operation that is appropriate for a case when there is a high nickel/(stainless steel) price ratio.

Table 5.8: User input for sample nickel-copper PSC computations (Feeds)

Matte	Ni:	20	wt%
	Co:	1	wt%
	Cu:	10	wt%
Flux	SiO_2:	85	wt%
	CaO:	5	wt%
	Al_2O_3:	5	wt%
	MgO:	5	wt%
Blast	Blast flowrate	45000	Nm^3/h
	Enrichment:	40	vol%O2
Ferronickel	Fe:	30	wt%
	Ni:	70	wt%
	Co:	0	wt%
	Cu:	0	wt%
	S:	0	wt%
	Si:	0	wt%
	Ca:	0	wt%
	Al:	0	wt%
	Mg:	0	wt%
	O:	0	wt%
	Density:	8.2	T/m^3
Specific Heat of Formation:		-126.2127	MJ/T
Heat Capacity Coefficient, A:		0.469	J/(g°C)
Heat Capacity Coefficient, B:		0	$J/(g(°C)^2)$
Heat Capacity Coefficient, C:		0	(J°C)/g
Heat Capacity Coefficient, D:		0	$J/(g(°C)^3)$

Table 5.9: User input for sample nickel-copper PSC computations (Temperatures)

Feed Temperatures		
Matte Feed Temperature:	1200	°C
Cold Feed Temperature:	30	°C
Blowing Temperatures		
Blast Temperature:	50	°C
Bath Temperature During Blow:	1050	°C (minimum)
Bath Temperature During Blow:	1250	°C (maximum)
Product Temperatures		
Slag Temperature:	1230	°C
Offgas Temperature During Slag Blow:	1200	°C
Final Discharge Temperature:	1150	°C (minimum)
Final Discharge Temperature:	1250	°C (maximum)

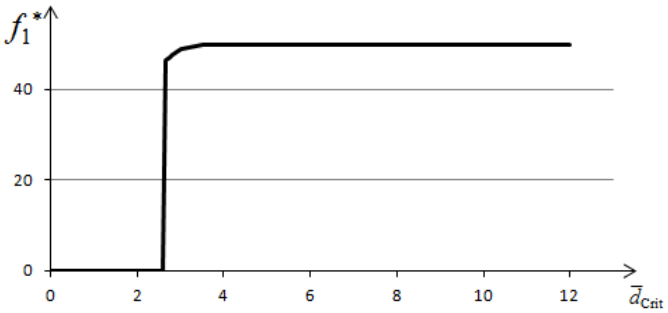
Comparing Table 5.6 to Table 5.1, the nickel-copper computations use larger ladle and bath volumes. Table 5.7 is comparable to Table 5.2, except that a global constraint has been implemented, to ensure that at least six ladles of matte are processed.

Table 5.8 describes typical matte and flux compositions for nickel-copper smelters. On the other hand, the oxygen enrichment is much higher than for typical smelters, and is characteristic of the ALSI [45]. The ferronickel data was compiled from several sources [84, 85, 86].

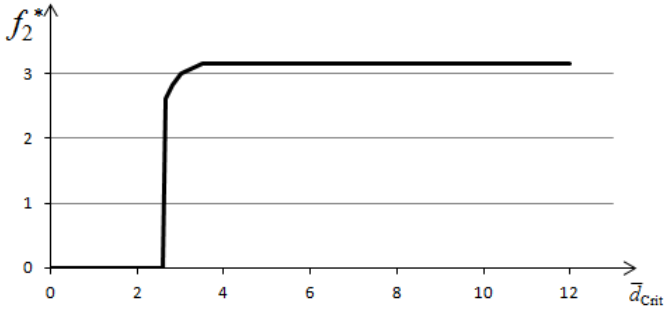
The computation time was found to be an average of 155.6 s. This is notably longer than the copper computations, but is still within a practical range. However, there were some numerical instabilities at $\overline{d}_{\text{Crit}} = 3.25$ h and $\overline{d}_{\text{Crit}} = 11.25$ h, which correspond to iterations 13 and 49, respectively. These iterations did not yield any solution, and are hence omitted from Figure 5.17. The optimal ratio objective was found to be 17.643 m^3 ferronickel / critical hour.

The solution is described by Figures 5.18-20 and Table 5.10. The first postcritical charging (of ferronickel) occurs at 2.63 h and causes a tremendous drop in temperature. This is followed by a sequence of blowing, charging and skimming, in which the maximum bath volume (160 m^3) is obtained exactly as the skimming temperature is obtained (1230°C); this synchronization of volume and temperature is somewhat sophisticated, as it combines MILP with Newton's Method (Equation 3.26). The final two skimming actions occur at volumes lower than 160 m^3, but the skimming temperature is still respected.

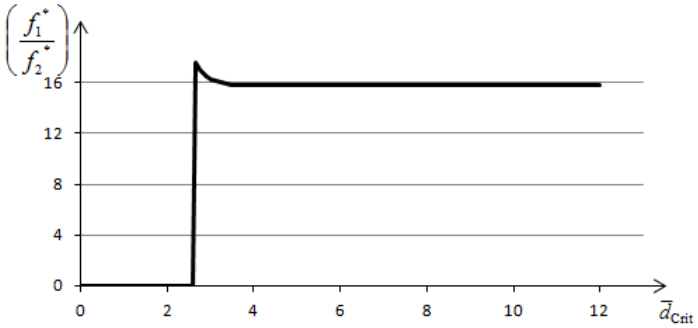
The MILP formulation has been successfully adapted to optimize single cycles. These results demonstrate the usefulness of MILP formulation for copper smelters, as well as nickel-copper smelters. The next step would be to develop industrial software that would schedule daily operations, using the MILP for the underlying computations.



(a)



(b)



(c)

Figure 5.17: Objective functions from sample nickel-copper PSC computations

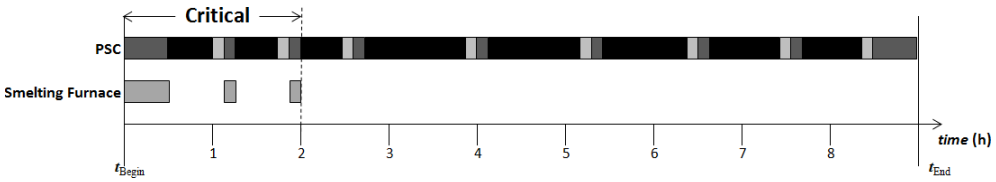


Figure 5.18: Optimal Gantt charts from sample nickel-copper PSC computations

Table 5.10: Feed tonnages from nickel-copper PSC computations

Transition	1	2	4	7	10	11	12	13	14
Start Time (h)	0	0.5	1.13	1.88	2.63	3.95	5.24	6.50	7.54
Finish Time (h)	0.5	1	1.25	2	3.95	5.24	6.50	7.54	8.45
Charging Feed									
Feed Matte	419.7	0	104.9	104.9	0	0	0	0	0
Flux	0	0	7.3	7.3	0	0	0	0	0
Ferronickel	40	0	6.6	6.7	86.8	81.5	77.0	49.2	31.9
Blowing Feed									
Flux	0	16.6	0	0	30.8	29.2	27.8	19.5	14.4

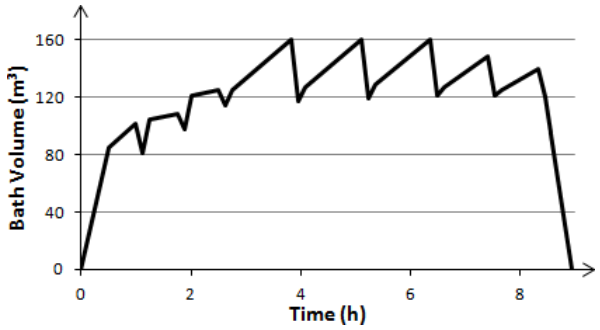


Figure 5.19: Bath volume from sample nickel-copper PSC computations

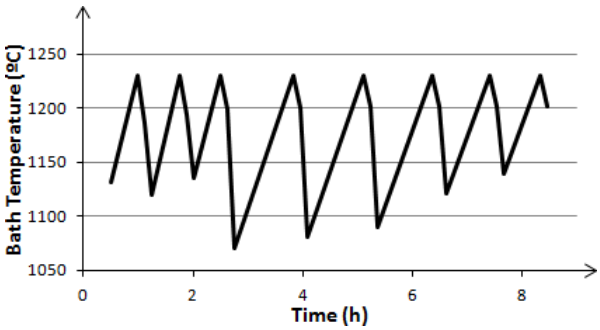


Figure 5.20: Bath temperature from sample nickel-copper PSC computations

CHAPTER 6

EXTENSIONS OF THE PSC MILP FORMULATION

6.1 Nonlinearity of the PS Converter Problem

6.1.1 Relaxation of the Complete-Discharge Condition

The current MILP formulation attains a high level of abstraction for the optimization of PS operations. However, there are two main directions for future work. One is to enhance the realism of the MILP by including nonlinear components, and the other is to generate customized solution for more rapid and extensive computations.

The MILP has compromised some basic nonlinear features, thus artificially contracting the solution space of the PSC Problem. From the modeling point-of-view, the main shortcomings are related to the complete-discharge condition (Subsection 4.4.2). Given that PS converting is a well-mixed reactor [7], it is reasonable to maintain the assumption of bath homogeneity, hence the concentration must be preserved even as the bath is being split. Without the complete-discharge condition, this assertion could not generally be satisfied by the MILP, due to the nonlinear relationships which will now be explained.

If transition l includes the discharging of a product stream $k \in \mathcal{Z}_{\text{NGProd}}$, then the total mass m_k^l is split into two substreams, the retained portion $m_{\text{Ret}k}^l$ and the discharged portion $(m_k^l - m_{\text{Ret}k}^l) > 0$, as depicted by Figure 6.1. Otherwise, if l does not include the discharging of k, then the entire stream is retained $m_k^l = m_{\text{Ret}k}$.

The homogeneity condition asserts that the same concentration must be held by both the retained and unretained portions. In terms of species,

$$w_{jk}^l = \frac{m_{j\text{Prod}}^l}{m_k^l} = \frac{m_{j\text{RetProd}}^l}{m_{\text{Ret}k}^l}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$, $j \in \mathcal{S}_{\text{k}}$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_{\text{o}}$. Cross-multiplication of the denominators gives

$$m_{j\text{Prod}}^l m_{\text{Ret}k}^l = m_{j\text{RetProd}}^l m_k^l$$

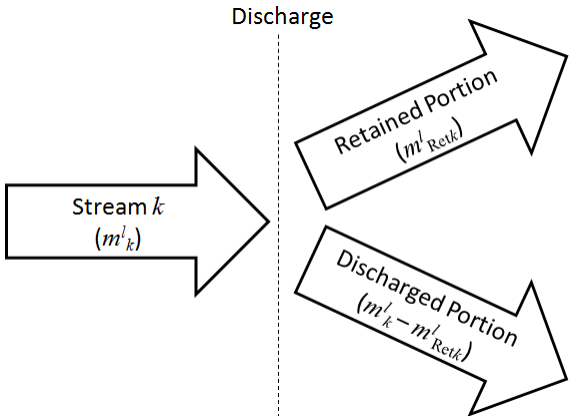


Figure 6.1: Splitting of a product stream during a discharge

In terms of the MILP variables,

$$m^l_{j\text{Prod}} \sum_{j' \in \mathcal{S}_k} m^l_{j'\text{RetProd}} = m^l_{j\text{RetProd}} \sum_{j' \in \mathcal{S}_k} m^l_{j'\text{Prod}} \tag{6.1}$$

for all $k \in \mathcal{Z}_{\text{NGProd}}$, $j \in \mathcal{S}_k$ and $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$. In combination with the discharge volume balance (Equation 4.71), this last constraint would be sufficient to uniquely solve for $m^l_{j\text{RetProd}}$. However, Equation 6.1 is bilinear, hence not supported by the MILP structure.

The implication of temperature homogeneity is even more complicated, since it would replace the equations of Subsection 4.3.6 with

$$h^{l-}_{\text{Ret}} + h^l_{\text{Ch}} + h^l_{\text{NGBlow}} + h^l_{\text{Blast}} - h^l_{\text{Offgas}} - \textstyle\sum_{i=1}^{6} h^l_{\text{Env}i}$$

$$= \tag{6.2}$$

$$\textstyle\sum_{k' \in \mathcal{Z}_{\text{NGFeed}}} \left[w_{\text{H}k'}(T^{\text{DCh},l})\right] m^l_{\text{Ret}k} + \sum_{j \in \mathcal{Z}_{\text{NGProd}}} \left[w_{\text{H}j}(T^{\text{DCh},l})\right] m^l_{j\text{Prod}}$$

and

$$h_{\text{DCh}} = \sum_{j \in \mathcal{S}_{\text{Prod}}} \left[w_{\text{H}j}(T^{\text{DCh},l})\right] \left(m^l_{j\text{Prod}} - m^l_{j\text{RetProd}}\right) \tag{6.3}$$

for all $l \in \mathcal{A}_{\text{PSC}} \setminus \mathcal{A}_\circ$, in which $T^{\text{DCh},l}$ is the bath temperature of obj(l) at time t^l_6. Equation 6.2 is a nonlinear equation that, in principle, can be solved by Newton's Method in order to determine the temperature $T^{\text{DCh},l}$ (See Subsection 3.2.4). Subsequently, Equation 6.3 asserts that this same temperature is held by the discharged products.

If temperature homogeneity (Equations 6.2-3) could somehow be implemented, then the

mathematical program would immediately accept temperature bounds for intermediate discharges. It would no longer be necessary that $\underline{T}^{\mathrm{DCh},k} = \overline{T}^{\mathrm{DCh},k} = T^{\mathrm{DCh},k}$ for all $k \in \mathcal{T}_{\mathrm{PSC,IDCh}}$, hence there would no longer be a formal distinction between intermediate and final discharges. Skimming temperatures could then vary, rather than artificially holding predetermined values. Ultimately, this would lead to a more flexible management of cold charges and fluxing agents, which are used partly for temperature control.

Furthermore, a nonlinear implementation could accommodate the imperfect splitting of streams. A skimming operation, for instance, is assumed to remove all of the slag, and only the slag. In practice, there may be some slag that would remain connected to the bath; conversely, some of the matte would be entrained into the outgoing slag. More effective skimming usually requires more time, or a different set of equipment. Thus, there are important studies to be made, regarding rough versus delicate skimming.

The imperfect splitting of streams is related to the product flows $\mathcal{F}_{\mathrm{Prod}}$. New members could be added into $\mathcal{F}_{\mathrm{Prod}}$ in order to represent the entrained flows. Much like the blending conditions presented in Subsection 4.5.1 for incoming feeds, there could be entrainment conditions that would tie a main outgoing flow with entrained secondary flows.

The complete-discharge condition remains an essential part of the formulation because it avoids the nonlinearity imposed by the homogeneity of concentration and temperature. However, the relaxation of the complete-discharge condition would offer more realism than the current MILP formulation. It may be worthwhile to experiment with nonlinear solvers [80], to analyze the computational impact of Equations 6.1-3.

6.1.2 Heat Transfer

The MILP formulation permits linear equalities of heat and mass, but not of temperature (Subsection 4.5.4). Thus the fundamentals of heat transfer cannot be directly represented by MILP formulations, but are represented indirectly, as empirical linear formulations that must be supported by industrial data. If future formulations were to include nonlinear components, then the heat transfer could be represented as a hybrid of empirical and fundamental considerations.

During a transition $l \in \mathcal{A}_{\mathrm{PSC}} \setminus \mathcal{A}_{\circ}$, the heat retained in the bath of $\mathrm{obj}(l)$ is given by

$$h^{\mathrm{obj}(l)}(t) = h^{l-}_{\mathrm{Ret}} + h^{\mathrm{obj}(l)}_{\mathrm{Ch}}(t) + h^{\mathrm{obj}(l)}_{\mathrm{NGBlow}}(t) + h^{\mathrm{obj}(l)}_{\mathrm{Blast}}(t) - h^{\mathrm{obj}(l)}_{\mathrm{Offgas}}(t) - h^{\mathrm{obj}(l)}_{\mathrm{DCh}}(t) - h^{\mathrm{obj}(l)}_{\mathrm{Env}}(t) \qquad (6.4)$$

for all $t \in (t^{l-}, t^{l}]$. This is a continuous extension of the forward heat computation (Equation

4.72), in which

- $h_{\mathrm{Ch}}^{\mathrm{obj}(l)}(t)$ is the accumulated heat fed into the bath of obj(l), as part of the charge of transition l
- $h_{\mathrm{NGBlow}}^{\mathrm{obj}(l)}(t)$ is the accumulated heat fed into the bath of obj(l), as part of the nongaseous blow feed of transition l
- $h_{\mathrm{Blast}}^{\mathrm{obj}(l)}(t)$ is the accumulated heat blown into the bath of obj(l), as part of the blast of transition l
- $h_{\mathrm{Offgas}}^{\mathrm{obj}(l)}(t)$ is the accumulated heat leaving the bath of obj(l), as part of the offgas of transition l
- $h_{\mathrm{DCh}}^{\mathrm{obj}(l)}(t)$ is the accumulated heat leaving the bath of obj(l), as part of the discharge of transition l
- $h_{\mathrm{Env}}^{\mathrm{obj}(l)}(t)$ is the accumulated environmental heat loss of obj(l), as part of transition l

These terms all have influence on the bath temperature $T^{\mathrm{obj}(l)}(t)$, which is hence an additional function of time $t \in (t^{l-}, t^{l}]$. In turn, the bath temperature influences the outgoing heats, $h_{\mathrm{Offgas}}^{\mathrm{obj}(l)}$, $h_{\mathrm{DCh}}^{\mathrm{obj}(l)}$ and $h_{\mathrm{Env}i}^{\mathrm{obj}(l)}$. The outgoing heats and the evolving bath temperature are coupled.

The semi-discrete treatment that was developed in Chapters 2 and 4, is largely satisfactory for the feed heats, $h_{\mathrm{Ch}}^{\mathrm{obj}(l)}$, $h_{\mathrm{NGBlow}}^{\mathrm{obj}(l)}$ and $h_{\mathrm{Blast}}^{\mathrm{obj}(l)}$, considering that these terms only vary during the corresponding action. For instance, the charging action occurs entirely in the second segment $(t_1^l, t_2^l]$, so that

$$h_{\mathrm{Ch}}^{\mathrm{obj}(l)}(t) = \begin{cases} 0 & \text{if } t^{l-} < t \leq t_1^l \\ \displaystyle\int_{t_1^l}^{t} \dot{h}_{\mathrm{Ch}}^{\mathrm{obj}(l)}(t')dt' & \text{if } t_1^l < t \leq t_2^l \\ h_{\mathrm{Ch}}^{l} & \text{if } t_2^l < t \leq t^l \end{cases} \tag{6.5}$$

in which $\dot{h}_{\mathrm{Ch}}^{\mathrm{obj}(l)}$ describes how the charging action is distributed over the duration of segment 2, and the integral can be interpreted in the sense of Lebesgue [87]. This distribution $\dot{h}_{\mathrm{Ch}}^{\mathrm{obj}(l)}$ can be determined by the feed temperatures of the charges, and the rates at which they are

fed into the converter. The blow feeds can be treated similarly, such that

$$h^{\mathrm{obj}(l)}_{\mathrm{NGBlow}}(t) = \begin{cases} 0 & \text{if } t^{l-} < t \leq t^l_3 \\ \int_{t^l_3}^{t} \dot{h}^{\mathrm{obj}(l)}_{\mathrm{NGBlow}}(t')dt' & \text{if } t^l_3 < t \leq t^l_4 \\ h^l_{\mathrm{NGBlow}} & \text{if } t^l_4 < t \leq t^l \end{cases} \tag{6.6}$$

$$h^{\mathrm{obj}(l)}_{\mathrm{Blast}}(t) = \begin{cases} 0 & \text{if } t^{l-} < t \leq t^l_3 \\ \int_{t^l_3}^{t} \dot{h}^{\mathrm{obj}(l)}_{\mathrm{Blast}}(t')dt' & \text{if } t^l_3 < t \leq t^l_4 \\ h^l_{\mathrm{Blast}} & \text{if } t^l_4 < t \leq t^l \end{cases} \tag{6.7}$$

such that the variations occur during segment 4. Subsection 4.3.4 considers blast compositions and rates that are prescribed by the transition type, such that

$$\dot{h}^{\mathrm{obj}(l)}_{\mathrm{Blast}} = \begin{cases} \dot{h}^k_{\mathrm{Blast}} & \text{if } l \text{ is of a type } k \in \mathcal{T}_{\mathrm{PSC}} \\ 0 & \text{otherwise} \end{cases}$$

However $\dot{h}^{\mathrm{obj}(l)}_{\mathrm{Blast}}(t)$ could now be allowed to vary during the blowing action $t \in (t^l_3, t^l_4]$.

In itself, the discharge heat is not so different than the feed heats,

$$h^{\mathrm{obj}(l)}_{\mathrm{DCh}}(t) = \begin{cases} 0 & \text{if } t^{l-} < t \leq t^l_5 \\ \int_{t^l_5}^{t} \dot{h}^{\mathrm{obj}(l)}_{\mathrm{DCh}}(T^{\mathrm{obj}(l)}(t'), t')dt' & \text{if } t^l_5 < t \leq t^l_6 \\ h^l_{\mathrm{DCh}} & \text{if } t^l_6 < t \leq t^l \end{cases} \tag{6.8}$$

However, the discharge heat rate $\dot{h}^{\mathrm{obj}(l)}_{\mathrm{DCh}}$ depends on the mass removal schedule, as well as the evolving temperature of the bath $T^{\mathrm{obj}(l)}$. This variable temperature is dependent on time, in turn, due to the environmental heat losses $h^{\mathrm{obj}(l)}_{\mathrm{Env}}$, described below. Equations 6.5-7 are less complicated than Equation 6.8 because feed temperatures are parameters, whereas the discharge temperatures are variable.

Even more complicated is the offgas convection $h^{\mathrm{obj}(l)}_{\mathrm{Offgas}}$, as the mass expulsion of the offgas

is a passive response to the converting reactions. At the onset of a blowing action, a quasi-steady state is rapidly obtained, whereby the offgas is expelled as it is being formed. The rate of offgas exhaust is not steady-state in a strict sense [61], because it is subject to operational and thermochemical parameters. Firstly, there may be changes in the blast flowrate or oxygen efficiency. Secondly, there may be changes in the reaction regime.

The reaction regime inside converter j can be represented using binary functions,

$$\beta_{\mathrm{Rg}k}^{\mathrm{PSC}j}(t) = \begin{cases} 1 & \text{if the bath of converter } j \text{ is in regime } k \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

for all $j \in \{1, 2, \ldots, n_{\mathrm{PSC}}\}$ and $k \in \mathcal{R}$. These functions are the regime determinants; they are a continuous extension of the binary variables presented in Subsection 4.3.3, such that $\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t_4^l) = \beta_{\mathrm{Rg}k}^{l}$. These regimes are subject to

$$\sum_{k\in\mathcal{R}} \beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t) = 1 \tag{6.9}$$

for $t \in (t_3^l, t_4^l]$, which has a similar role as Equation 4.44. As before, the regime determinants are not well defined when there are no product streams. To develop the $h_{\mathrm{Offgas}}^{\mathrm{obj}(l)}$ term, the current discussion focuses only on blowing transition, i.e. l such that $t_4^l > t_3^l$, hence some product streams are presumably present in obj(l).

The regime determinants $\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t)$ are related to the product masses. For $t \in (t_3^l, t_4^l]$ and $i \in \mathcal{E}$, the amount of i present in the nongaseous streams is denoted $m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}(t)$. This is like a continuous extension of $m_{i\mathrm{Prod}}^{l}$ (see Equations 4.35-36), except that $m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}$ focuses only on the nongaseous products,

$$\begin{aligned} m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}(t) \;=\; & \sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l-} + \sum_{j\in\mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{RetProd}}^{l-} + \sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} \int_{t_1^l}^{t} \dot{m}_k^{\mathrm{obj}(l)}(t')dt' \\ & + \int_{t_3^l}^{t} \left[e_{\mathrm{O}} \dot{m}_{\mathrm{O,Blast}}^{\mathrm{obj}(l)}(t')\right] \left(\sum_{k\in\mathcal{R}} r_i^k \left[\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t')\right]\right) dt' \end{aligned} \tag{6.10}$$

where $\dot{m}_{\mathrm{O,Blast}}^{\mathrm{obj}(l)}$ is the mass flowrate of blast oxygen, r_i^k is the change in $m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}$ per mass of reacted blast oxygen under regime k, and $\dot{m}_k^{\mathrm{obj}(l)}$ is the mass feed distribution of stream k.

It is assumed that all the nongeasous feed streams are rapidly incorporated into the

products, so that

$$\lim_{t\to t_3^l} m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}(t) = \sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} m_{\mathrm{Ret}k}^{l-} + \sum_{j\in\mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{RetProd}}^{l-} + \sum_{k\in\mathcal{Z}_{\mathrm{NGFeed}}} w_{ik} \int_{t_1^l}^{t_2^l} \dot{m}_k^{\mathrm{obj}(l)}(t')dt'$$

which is a continuous extension of the complete-reaction condition presented in Subsection 4.4.1. Prior to blowing, the feed stream changes are due entirely to charging, which is limited to the interval $(t_1^l, t_2^l]$.

Sulfur and oxygen are the only bath elements that are exhausted as offgas. (Nitrogen is not a bath element, as it is not retained). Therefore, $r_i^k = 0$ for all $i \in \mathcal{E} \setminus \{\mathrm{S,O}\}$, and Table 6.1 contains values only for r_{S}^k and r_{O}^k. The Slag-Blow is complicated by the simultaneous formation of fayalite and of magnetite (Subsection 1.3.2); it can be shown that,

$$r_{\mathrm{O}}^{\mathrm{SlagBlow}} = \frac{M_{\mathrm{Fe_3O_4}} r_{\mathrm{FS}} + 4M_{\mathrm{FeO}}}{M_{3\mathrm{Fe_3O_4}} r_{\mathrm{FS}} + 10M_{\mathrm{FeO}}} \tag{6.11}$$

$$r_{\mathrm{S}}^{\mathrm{SlagBlow}} = -\left(\frac{M_{\mathrm{S}}}{M_{\mathrm{O}}}\right)\left(\frac{M_{\mathrm{Fe_3O_4}} r_{\mathrm{FS}} + 3M_{\mathrm{FeO}}}{M_{3\mathrm{Fe_3O_4}} r_{\mathrm{FS}} + 10M_{\mathrm{FeO}}}\right) \tag{6.12}$$

which uses the ferroslag ratio r_{FS} that had been introduced in Subsection 3.4.2 as a constant, although it is conceivable that r_{FS} would vary over time, given the evolving slag chemistry [32], etc. The remaining values of Table 6.1 can be determined from the stoichiometries presented in Subsection 1.3.2; for example, the Nickel-Overblow results 6 moles of NiO for every 7 moles of reacted blast O_2, so that

$$r_{\mathrm{O}}^{\mathrm{NickelOverblow}} = \left(\frac{6M_{\mathrm{NiO}}}{7M_{\mathrm{O_2}}}\right) w_{\mathrm{O,NiO}} = 0.42857$$

Similar calculations can be applied for the rest of Table 6.1.

Equation 6.10 gives the elemental mass composition of the bath at time $t \in (t_3^l, t_4^l]$,

Table 6.1: Values for r_{S}^k and r_{O}^k for the different reaction regimes

Reaction Regime (k)	r_{O}^k	r_{S}^k
SlagBlow	(Equation 6.10)	(Equation 6.11)
NickelOverblow	0.42857	-0.57261
CobaltOverblow	0.33333	-0.66805
CopperBlow	0	-1.0021
CopperOverblow	1	0

which determines the regime. Thus $\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t)$ must be computed simultaneously to $m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}(t)$. Firstly, there are several nongaseous species that are eliminated in a manner similar to Equation 4.42,

$$m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t) \leq \overline{m}_{j\mathrm{Prod}} \sum_{k \in \mathcal{R}_j} \beta_{\mathrm{Rg}k}^{\mathrm{PSC}j}(t) \tag{6.13}$$

for all $j \in \mathcal{S}_{\mathrm{Rg}}$ and $t \in (t_3^l, t_4^l]$; for all $j \in \mathcal{S}_{\mathrm{NGProd}} \supset \mathcal{S}_{\mathrm{Rg}}$, $m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t)$ is the amount of species j in obj(l). Subsequently, the nongaseous product speciation balance is given by

$$\sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} w_{ij} m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t) = m_{i\mathrm{NGProd}}^{\mathrm{obj}(l)}(t) \tag{6.14}$$

for $t \in (t_3^l, t_4^l]$ and $i \in \mathcal{E}$. The product speciation balance of Equations 3.37 and 4.43 considered all of the product species $\mathcal{S}_{\mathrm{Prod}}$, whereas Equation 6.14 considers only the nongaseous product species $\mathcal{S}_{\mathrm{NGProd}}$. For a continuous treatment, the offgas species are considered separately, as described below.

Equations 6.9-14 combine with the first two columns of Table 6.1, and the nonnegativity of $m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}$, to give appropriate values for $\beta_{\mathrm{Rg}k}^{\mathrm{PSC}j}(t)$. A forward-looking algorithm can be devised to detect the times when a regime change would occur. The algorithm may find that the regime would change at some time t_{RgChange}; if $t_{\mathrm{RgChange}} < t_4^l$ then the change occurs within the current blowing action, and the algorithm should be applied again to determine if there would be yet another regime change.

The flow of offgas species is described by

$$\dot{m}_{\mathrm{O_2,Prod}}^{\mathrm{obj}(l)}(t) = (1 - e_{\mathrm{O}}) \left[\dot{m}_{\mathrm{O,Blast}}^{\mathrm{obj}(l)}(t) \right] \tag{6.15}$$

$$\dot{m}_{\mathrm{N_2,Prod}}^{\mathrm{obj}(l)}(t) = \dot{m}_{\mathrm{N,Blast}}^{\mathrm{obj}(l)}(t) \tag{6.16}$$

$$\dot{m}_{\mathrm{SO_2,Prod}}^{\mathrm{obj}(l)}(t) = \frac{\left[e_{\mathrm{O}} \dot{m}_{\mathrm{O,Blast}}^{\mathrm{obj}(l)}(t) \right]}{w_{\mathrm{S,SO_2}}} \sum_{k \in \mathcal{R}} \left(-r_{\mathrm{S}}^k \right) \left[\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}(t) \right] \tag{6.17}$$

for $t \in (t_3^l, t_4^l]$, in which $\dot{m}_{\mathrm{O,Blast}}^{\mathrm{obj}(l)}$ and $\dot{m}_{\mathrm{N,Blast}}^{\mathrm{obj}(l)}$ are both determined from industrial blast parameters, considering a two-element blast (Subsection 3.3.3). Thus $\dot{m}_{j\mathrm{Prod}}^{\mathrm{obj}(l)}$ is the rate at which species j is produced in obj(l). In particular, Equation 6.17 depends on the regime determinants $\beta_{\mathrm{Rg}k}^{\mathrm{obj}(l)}$ computed from Equations 6.9-6.14, and the rate of sulfur expulsion r_{S}^k that is available through Equation 6.12 and Table 6.1.

Finally, the offgas convection is computed as

$$h^{\mathrm{obj}(l)}_{\mathrm{Offgas}}(t) = \begin{cases} 0 & \text{if } t^{l-} < t \leq t^l_3 \\ \displaystyle\sum_{j \in \mathcal{S}_{\mathrm{Offgas}}} \int_{t^l_3}^{t} \left[w_{\mathrm{H}j}(T^{\mathrm{obj}(l)}(t'), t')\right] \left[\dot{m}^{\mathrm{obj}(l)}_{j\mathrm{Offgas}}(t)\right] dt' & \text{if } t^l_3 < t \leq t^l_4 \\ h^l_{\mathrm{Offgas}} & \text{if } t^l_4 < t \leq t^l \end{cases} \tag{6.18}$$

for $t \in (t^{l-}, t^l]$, in which $w_{\mathrm{H}j}$ is computed as described in Subsection 3.2.3. Thus the computation of Equation 6.18 depends on Equations 6.15-17 to compute the offgas flowrates, of which Equation 6.17 is dependent on Equations 6.9-6.14 to obtain the reaction regimes.

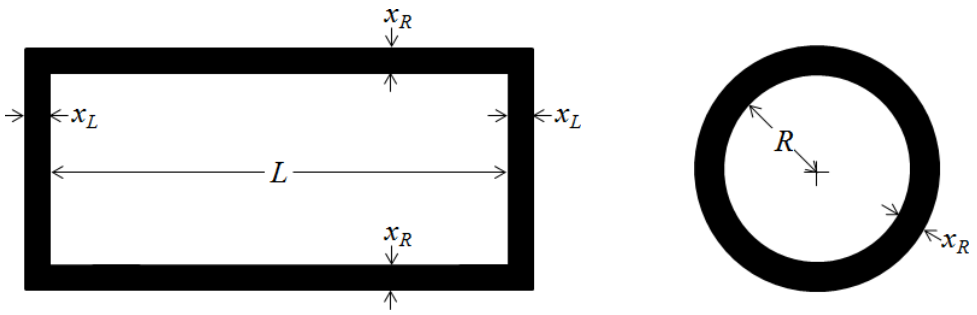
In the treatment of Subsection 4.3.7, the environmental heat losses $h^{\mathrm{obj}(l)}_{\mathrm{Env}}(t)$ are represented as piecewise linear functions, with possible discontinuities that may occur at the beginning and end of each segment $i \in \{1, 2, \ldots, 7\}$; the discontinuities and slopes were determined by the transition types.

A more fundamental approach [39, 52] considers thermal conduction through the walls of the converter, and radiation through the mouth. Following these considerations,

$$\begin{aligned} h^{\mathrm{obj}(l)}_{\mathrm{Env}}(t) = & \sum_{k \in \mathcal{T}_{\mathrm{PSC}}} h^k_{\mathrm{Env}\circ} \beta^{l-}_{\mathrm{Type}k} + \sum_{i=1}^{7} \left(\int_{t^{l-}}^{t} \delta(t' - t^l_i) dt' \right) \left(\sum_{k \in \mathcal{T}_{\mathrm{PSC}}} h^k_{\mathrm{Env}i} \beta^l_{\mathrm{Type}k} \right) \\ & + \kappa_1 G^{\mathrm{obj}(l)} \int_{t^{l-}}^{t} \left([T^{\mathrm{obj}(l)}(t')] - [T^{\mathrm{obj}(l),\mathrm{Ext}}(t')] + \frac{\kappa_2}{2} \left([T^{\mathrm{obj}(l)}(t')]^2 - [T^{\mathrm{obj}(l),\mathrm{Ext}}(t')]^2 \right) \right) dt' \\ & + \sigma A_{\mathrm{Mouth}} [\varepsilon(t)] \int_{t^{l-}}^{t} \left([T^{\mathrm{obj}(l)}(t')]^4 - [T^{\mathrm{obj}(l),\mathrm{Target}}(t')]^4 \right) dt' \end{aligned} \tag{6.19}$$

for $t \in (t^{l-}, t^l]$, in which κ_1 and κ_2 are the first and second conductivity parameters of the refractory [52], σ is the Stefan-Boltzman radiation constant [88], ε is the emissivity of the bath surface, A_{Mouth} is the area of the converter mouth, $T^{\mathrm{obj}(l),\mathrm{Ext}}$ is the temperature of the exterior walls of obj(l), and $T^{\mathrm{obj}(l),\mathrm{Target}}$ is temperature of the radiation target, and $G^{\mathrm{obj}(l)}$ is a geometrical factor that is described below. The δ is the Dirac delta distribution [89], which is used to incorporate the $h^k_{\mathrm{Env}i}$ parameters that had been introduced in Subsection 4.3.7.

The conductivity term includes the factor $G^{\mathrm{obj}(l)}$, which is based on the cylindrical geom-



(a) Side-view (b) Cross-section

Figure 6.2: Simplified geometry of a converter that has no mouth

etry on the converter (Figure 6.2). It can be taken as

$$G^{\mathrm{obj}(l)} = 2\left(\frac{\pi\left(R^{\mathrm{obj}(l)}\right)^2}{x_{\mathrm{L}}^{\mathrm{obj}(l)}} + \frac{L^{\mathrm{obj}(l)}}{\ln\left(1+\dfrac{x_{\mathrm{R}}^{\mathrm{obj}(l)}}{R^{\mathrm{obj}(l)}}\right)}\right) \tag{6.20}$$

in which $R^{\mathrm{obj}(l)}$, $L^{\mathrm{obj}(l)}$, $x_{\mathrm{R}}^{\mathrm{obj}(l)}$, and $x_{\mathrm{R}}^{\mathrm{obj}(l)}$ are the radius, length, radial thickness and endwall thickness of obj(l), respectively. This expression can be deduced by considering Fourier's law in cylindrical coordinates [90]. However, Equation 6.20 considers a simplified geometry in which the converter mouth is closed off (Figure 6.2). Nonetheless, industrial measurements have shown that Equation 6.20 is a valid approximation [39], considering that the radiation term dominates any fictitious mouth conduction.

Equation 6.19 contains three functions of time that can be further developed. Firstly, the exterior wall temperature $T^{\mathrm{obj}(l),\mathrm{Ext}}$ may be related to the ambient smelter temperature via Newton's Law of Cooling [91], although this would require some knowledge of the heat transfer coefficients. Secondly, the emissivity ε can be expanded in terms of the product masses, considering that the bath surface can be inhabited by matte, blister copper, or various types of slag. Also, $T^{\mathrm{obj}(l),\mathrm{Target}}$ may be equal to the hood temperature, or the ambient temperature, depending on whether or not the hood is lowered over the mouth.

All of the terms in the heat balance (Equation 6.4) have been described to some degree. Following this balance, the heat is distributed into the bath components,

$$h^{\mathrm{obj}(l)}(t) = \sum_{k' \in \mathcal{Z}_{\mathrm{NG}}} \left[w_{\mathrm{H}k'}(T^{\mathrm{obj}(l)}(t), t)\right]\left[m_{k'}^{\mathrm{obj}(l)}(t)\right]$$

or equivalently,

$$h^{\mathrm{obj}(l)}(t) = \sum_{k \in \mathcal{Z}_{\mathrm{NGFeed}}} \left[w_{\mathrm{H}k}(T^{\mathrm{obj}(l)}(t), t) \right] \left[m_k^{\mathrm{obj}(l)}(t) \right] + \sum_{j \in \mathcal{S}_{\mathrm{NGProd}}} \left[w_{\mathrm{H}j}(T^{\mathrm{obj}(l)}(t), t) \right] \left[m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t) \right] \tag{6.21}$$

for $t \in (t^{l-}, t^l]$, in which the feed streams masses are generally given by

$$m_k^{\mathrm{obj}(l)}(t) = \begin{cases} m_{\mathrm{Ret}k}^{l-} & \text{if } t^{l-} < t \leq t_1^l \\ m_{\mathrm{Ret}k}^{l-} + \displaystyle\int_{t_1^l}^{t} \dot{m}_k^{\mathrm{obj}(l)}(t')dt' & \text{if } t_1^l < t \leq t_3^l \\ 0 & \text{if } t_3^l < t_4^l \text{ and } t_3^l < t \leq t^l \\ m_k^l & \text{if } t_3^l = t_4^l \text{ and } t_3^l < t \leq t^l \end{cases} \tag{6.22}$$

for $k \in \mathcal{Z}_{\mathrm{NGFeed}}$, and the product species masses are given by

$$m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t) = \begin{cases} m_{j\mathrm{RetProd}}^{l-} & \text{if } t^{l-} < t \leq t_3^l \\ m_{j\mathrm{RetProd}}^{l-} + \displaystyle\int_{t_3^l}^{t} \dot{m}_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t')dt' & \text{if } t_3^l < t \leq t_4^l \\ m_{j\mathrm{Prod}}^{l} & \text{if } t_4^l < t \leq t_5^l \\ m_{j\mathrm{Prod}}^{l} + \displaystyle\int_{t_5^l}^{t} \dot{m}_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t')dt' & \text{if } t_5^l < t \leq t_6^l \\ m_{j\mathrm{RetProd}}^{l} & \text{if } t_6^l < t \leq t^l \end{cases} \tag{6.23}$$

for $j \in \mathcal{S}_{\mathrm{NGProd}}$. Equations 6.22-23 describe the various segments in which the bath masses vary, and those in which they do not vary; the latter makes use of the MILP variables. Equation 6.22 implements the complete-reaction, hence the distinction between blowing transitions ($t_3^l < t_4^l$) and nonblowing transitions ($t_3^l = t_4^l$). When l is a blowing transition, $m_{j\mathrm{Prod}}^{\mathrm{obj}(l)}(t)$ should be computed according to Equations 6.9-14 for $t \in (t_3^l, t_4^l]$, which is more precise than Equation 6.23.

The analysis described by Equations 6.4-23 cannot be directly incorporated into the MILP because of the nonlinear coupling that temperature has with the heat and mass variables.

However, the MILP can serve as a surrogate that would be grafted into an adaptive formulation [92]. Algorithms can be designed, which would execute preliminary MILP computations, then the nonlinear formulation would guide the alteration of the MILP parameters, followed by more MILP computations, and so on. Such a computational approach could be of interest for future work.

6.2 PS Operations Research

6.2.1 From Mathematical Programming to Advanced Algorithm Design

The results of Section 5.3 are a proof-of-concept for the MILP implementation, using a multipurpose solver (Subsection 5.2.1), based on multipurpose algorithms (Appendix B). Given this initial success, some effort should be devoted to hybridize these algorithms, hence to utilize the particular structure of the PSC Problem.

There is an important distinction between mathematical programming and algorithm design. Mathematical programming applies pre-existing algorithms that were conceived for abstract algebraic formulations; the current work has relied on the Simplex Method in conjunction with the Branch-and-Cut Method (Appendix B), as well as Newton's Method and the Active Set Method (Chapter 3). Beyond this, there has only been a cursory effort devoted to algorithm design; Chapter 5 was but a first attempt at the SC-PSC Problem, albeit a successful one.

The SC-PSC algorithm can definitely be improved. The first sweep should not perform 48 iterations, nor should the iterations be performed in a predetermined order. A more strategic sampling might balance early results for the two linear objectives, (Equations 5.19 and 5.21) hence focusing on the most promising ranges of $\overline{d}_{\text{Crit}}$.

The SC-PSC Problem was well motivated because the solutions can be incorporated into more general algorithms for PSC problems, as discussed in Subsections 5.1.2 and 5.1.3. The single cycles of the SC-PSC Problem can be assembled into a global schedule, using greedy algorithms [74]. Such a construction will not generally be optimal for the multi-converter PSC Problem, but it will provide an aggressive lower bound, which would accelerate the branch-and-cutting of the general PSC Problem.

An advanced algorithm could be based on the Gantt structure that was introduced in Section 2.1. Figure 6.3 demonstrates that there are two aspects to the Gantt structure, namely the geometry that is described by d^l and t^l, and the topology described by the categorical Type^l variables. In particular, the Type^l variables are implemented into the MILP as binary

variables $\beta^{l}_{\text{Type}k}$. Secondly, the topological dependencies across the different Gantt rows are described by $\beta^{lk}_{\text{Suppl}'k'}$. Rather than implementing the assignment types and dependencies as binary variables, they could be part of a filtering mechanism that would reject infeasible Gantt topologies; the Simplex method would only be performed on the most promising Gantt topologies. Such an approach would be characteristic of Constraint Programming [93], and may incorporate SC-PSC computations to direct the branching process.

Mathematical programming can lead to breakthroughs for the optimization of converter operations. The current MILP formulation is a robust backdrop from which algorithms and software will be developed, which will form the basis of PS operations research.

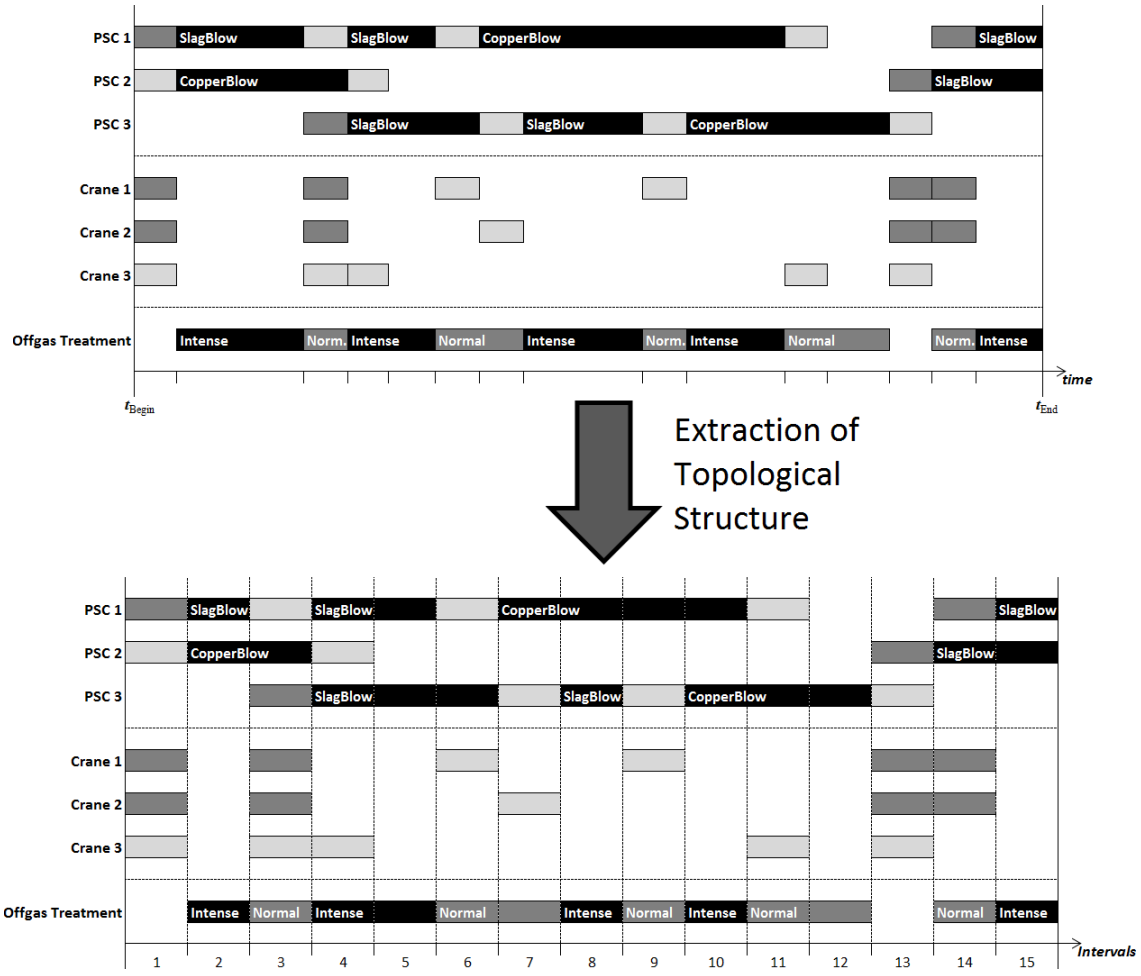


Figure 6.3: Extraction of Topological Information from a Gantt Chart

6.2.2 Fomenting Innovation

Nonferrous pyrometallurgy is a conservative industry, insomuch as PS converting has remained largely unchanged for over a hundred years [42]. Although operators and engineers may recognize opportunities for improvement, innovations come slowly, as it is difficult to quantify their benefits. Future adaptations of the MILP will help justify changes to smelters, and mitigate the tendency for technological stagnation.

The current MILP formulation is already being marketed to both the copper and nickel industries. The project has attracted industrial partners, which are now eager for future development. They have understood that any advancement to the MILP formulation could potentially improve the operation of all conventional copper and nickel smelters.

Schedule automation is a most welcome service that is now being offered to copper and nickel producers, as an adaptation of the MILP formulation. This is establishing partnerships between academia and industry, and providing valuable insight for customized algorithm development, as well as realistic plant data.

Smelters are driven to automate production scheduling, as it is intimately linked to costs and profits. It is not simply a question of the man-hours that have been devoted to traditional manual approaches. Scheduling algorithms can be extended to become general decision-making software for smelter operations. It becomes convenient to simulate several different strategies, and to analyze their impacts on the converting aisle, as well as the related ancillary equipment, and the entire production chain.

Established smelters are unlikely to accept changes to their daily operations, unless these changes can be supported by actual operational data. Given the stochastic nature of this data, a major change may require the simulation of hundreds or thousands of days, hence to explore the distribution of outcomes. This type of simulation is only practical if the scheduling algorithms can be automated within the simulation; otherwise, hundreds of man-hours would be required to compile schedules that would be manually fed into the simulation. Thus the deterministic MILP formulation should be incorporated into stochastic simulations, in order to justify alterations of the smelter.

The Peirce-Smith MILP formulation is itself an important innovation for the copper and nickel industries. Moreover, the formulation will facilitate other innovations, as the MILP formulation quantifies the potential benefits of these future works.

REFERENCES

[1] J. Kapusta,"JOM World Nonferrous Smelter Survey Part I: Copper", *Journal of Metals*, 56 (7) (2004), 21-27.

[2] A. Warner, C. Diaz, A. Dalvi, P. Mackey and A. Tarasov,"JOM World Nonferrous Smelter Survey Part III: Nickel Laterite", *Journal of Metals*, 58 (4) (2006), 11-20.

[3] A. Warner, C. Diaz, A. Dalvi, P. Mackey, A. Tarasov and R. Jones,"JOM World Nonferrous Smelter Survey Part VI: Nickel Sulphide", *Journal of Metals*, 59 (4) (2007), 58-72.

[4] D. Edelstein, "Copper", *Mineral Commodity Summaries*, U.S. Geological Survey (2013), 48-49.

[5] P. Kuck, "Nickel", *Mineral Commodity Summaries*, U.S. Geological Survey (2013), 108-109.

[6] H. Dettmer, "Introduction to the Theory of Constraints", Chapter 1 of *The Logical Thinking Process: A Systems Approach to Complex Problem Solving*, ASQ Quality Press (2007), 3-30.

[7] W. Davenport, M. King, M. Schlesinger and A. Biswas, "Batch Converting of Copper Matte", Chapter 9 of *Extractive Metallurgy of Copper*, Pergamon (2002), 131-154.

[8] R. Moskalyk and A. Alfantazi, "Review of Copper Pyrometallurgical Practice: Today and Tomorrow", Journal of Mineral Engineering, 16 (10) (2003), 898-919.

[9] F. Crundwell, M. Moats, V. Ramachandran, T. Robinson and W. Davenport, "Converting - Final Oxidation of Iron From Molten Matte", Chapter 19 of *Extractive Metallurgy of Nickel, Cobalt and Platinum Group Metals*, Elsevier (2011), 233-246.

[10] K. Ng, J. Kapusta, R. Harris, A. Wraith and R. Parra,"Modeling Peirce-Smith Converter Operating Costs", *Journal of Metals*, 57(7)(2005), 52-57.

[11] N. Tripathi, P. Coursol, D. Tisdale and P. Mackey, "Advanced Metallurgical Modeling of Ni-Cu Smelting at the Xstrata Nickel's Sudbury Smelter", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 251-261.

[12] P. Coursol and P. Mackey, "Optimization of the Xstrata Copper-Horne Smelter Operation Using Discrete Event Simulation", *Canadian Institute of Mining, Metallurgy and Petroleum Bulletin*, 102 (1114) (2009), 5-10.

[13] J. Stewart, "Newton's Method", Section 4.7 of *Calculus: Concepts and Contexts*, Cengage Learning (2009), 312-315.

[14] P. Armstrong, "The CDIO Syllabus: Learning Outcomes for Engineering Education", Chapter 3 of *Rethinking Engineering Education*, Springer (2007), 45-76.

[15] C. Gill,"Nonreactive Metals Pyrometallurgical Treatments", Chapter 1 of *Nonferrous Extractive Metallurgy*, John Wiley & Sons (1980), 7-140.

[16] C. Gill,"Nonreactive Metals Hydrometallurgical Treatments", Chapter 2 of *Nonferrous Extractive Metallurgy*, John Wiley & Sons (1980), 141-203.

[17] "Smelt", Online Etymology Dictionary, *www.etymonline.com/index.php?term=smelt* , accessed November 23rd 2011.

[18] G. Kordosky, "Copper Recovery Using Leach/Solvent Extraction/Electrowinning Technology: Forty Years of Innovation, 2.2 Million Tonnes of Copper Annually", *Journal of the South African Institute of Mining and Metallurgy*, Nov-Dec (2002), 445-450.

[19] F. Crundwell, M. Moats, V. Ramachandran, T. Robinson and W. Davenport, "Hydrometallurgical Production of High Purity Nickel and Cobalt", Chapter 23 of *Extractive Metallurgy of Nickel, Cobalt and Platinum Group Metals*, Elsevier (2011), 281-300.

[20] H. Hasan, "How Aluminum Is Produced", Chapter 5 of *Aluminum*, Rosen Publishing (2007), 33-36.

[21] M. Renner, "World Metals Production Surges", *Vital Signs*, Worldwatch Institute (2009), 1-6.

[22] "Nickel", Chapter 32 of *Canadian Minerals Yearbook*, National Research Counsil of Canada (2005), 1-32.

[23] C. Gill, introductory chapter of *Nonferrous Extractive Metallurgy*, John Wiley & Sons (1980), 1-5.

[24] X. Xiao, X. Songwen, G. Xueyi, H. Kelong and Y. Ryochi, "LCA Case Study of Zinc Hydro and Pyro-Metallurgical Process in China", *International Journal of Life Cycle Assessment*, 8 (3) (2003), 151-155.

[25] H. Hasan, "History of Aluminum", Chapter 1 of *Aluminum*, Rosen Publishing (2007), 6-9.

[26] I. Kojo, M. Lahtinen and E. Miettinen, "Flash Converting - Sustainable Technology Now and in the Future", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 383-395.

[27] F. Crundwell, M. Moats, V. Ramachandran, T. Robinson and W. Davenport, "Slow Cooling and Solidification of Converter Matte", Chapter 21 of *Extractive Metallurgy of Nickel, Cobalt and Platinum Group Metals*, Elsevier (2011), 259-268.

[28] J. Shirley, "Hydromet Development and Commercial Nickel Processing Plant Update", presentation, (Newfoundland Branch of *Canadian Institute of Mining, Metallurgy and Petroleum*, 2006).

[29] L. Southwick, "William Peirce and E. A. Cappelen Smith and their Amazing Copper Converting Machine", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 3-27.

[30] W. Peirce and E. Smith, "Method of and Converter for Bessemerizing Copper Matte", Patent 942342, United States Patent Office (1909).

[31] T. Miller, J. Jimenez, A. Sharan and D. Goldstein, "Oxygen Steelmaking Process", Chapter 9 of *The Making, Shaping and Treating of Steel*, AISE Steel Foundation (1998), 475-524.

[32] C. Ramirez, P. Ruz, G. Riveros, A. Warczok and R. Treimer, "Chrome-Magnesite Refractory Corrosion with Olivine Slag of High Cuprous Oxide Content", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 71-80.

[33] W. Davenport, M. King, M. Schlesinger and A. Biswas, "Copper Loss in Slag", Chapter 11 of *Extractive Metallurgy of Copper*, Pergamon (2002), 173-185.

[34] A. Sundström, J. Eksteen and G. Georgalli, "A Review of the Physical Properties of Base Metal Mattes", *Journal of the South African Institute of Mining and Metallurgy*, 108 (2008), 431-448.

[35] P. Voigt, B. Hogg and O. Pasca, "Application of Semtech Optical Process Control System at Mount Isa Mines' Copper Converters", *First Extractive Metallurgy Operators' Conference* (2005), 1-9.

[36] T. Prietl, A. Filzwieser and S. Wallner, "Productivity Increase in a Peirce-Smith Converter Using the COP KIN and OPC System", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 177-190.

[37] P. Coursol, B. Davis, A. Roy and M. Lebel,"Oxygen Removal from Blister Copper by Copper Oxide Formation", *Journal of Metals*, 57(7)(2005), 64-67.

[38] Photograph of newly installed Peirce-Smith converter at the Harjavalta Smelter, Kopar Oy, *www.kopar.fi/en/products/drums/converter.html*, accessed November 23rd 2011.

[39] A. Kyllo, G. Richards and S. Marcuson, "A Mathematical Model of the Nickel Converter: Part 2. Application and Analysis of Converter Operation", *Metallurgical Transactions B*, 23 (1992), 574-582.

[40] S. Chen, H. Mansikkaviita, M. Rythonen and I. Kylmäkorpi, "Continuous Improvement in Peirce-Smith Converter Design - Kumera's Approach", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 361-365.

[41] C. Goñi, M. Barbés, V. Bazán, E. Brandaleze, R. Parra and L. González, "The Mechanism of Thermal Spalling in the Wear of the Peirce-Smith Copper Converter", *Journal of the Ceramic Society of Japan*, 114 (8) (2009), 672-675.

[42] T. Price, C. Harris, S. Hills, W. Boyd and A. Wraith, "Peirce-Smith Converting: Another 100 Years?", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 181-197.

[43] D. Treilhard, "The Impact of Economic and Metallurgical Change in Copper Smelter Design", *Western Miner*, 45 (1-6) (1972), 34-47.

[44] A. Bustos, M. Cardoen and B. Janssens, "High Oxygen Enrichment at UM-Hoboken Converters", *Pyrometallurgy of Copper* (Copper 95 - Cobre 95 International Conference, 1995), 255-269.

[45] J. Kapusta, H. Stickling and W. Tai, "High Oxygen Shrouded Injection At Falconbridge: Five Years of Operation", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 47-60.

[46] B. Salt and E. Cerilli, "Evolution of the Converter Aisle at Xstrata Nickel's Sudbury Smelter", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 135-149.

[47] L. Pradenas, J. Zúñiga and V. Parada, "CODELCO Chile Programs its Copper-Smelting Operations", *Interfaces*, 36 (4) (2006), 296-301.

[48] L. Mailleret and V. Lemesle, "A note on semi-discrete modeling in the life sciences", Royal Society (2009), 1-21.

[49] S. Raczynski, "Semi-discrete events and models in categorical language", *International Journal of Simulation Modeling*, 11 (2) (2012), 89-96.

[50] A. Bryman and D. Cramer, "Constructing Variables", Chapter 2 of *The Handbook of Data Analysis*, SAGE (2009), 17-34.

[51] M. Zhou and K. Venkatesh, "Fundamentals of Petri Nets", Chapter 4 of *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*, World Scientific (2000), 59-90.

[52] A. Kyllo and G. Richards, "A Mathematical Model of the Nickel Converter: Part I. Model Development and Verification", *Metallurgical Transactions B*, 22 (1991), 153-161.

[53] A. Navarra and J. Kapusta, "Decision-making Software Development for Incremental Improvement of Nickel Matte Conversion", *Pyrometallurgy of Nickel and Cobalt*, (Conference of Metallurgists, 2009), 611-619.

[54] K. Davidson, K. Davidson and A. Donsig, "Convexity and Optimization", Chapter 16 of *Real Analysis and Applications: Theory in Practice*, Springer (2010), 449-504.

[55] A. Navarra, A. Pubill and J. Kapusta, "Convex Projection to Estimate Heat Content of Cold Charges in Peirce-Smith Converting", *Computational Thermodynamics and Kinetics* (The Minerals, Metals and Materials Society Conference, 2012), 151-158.

[56] C. Bale, P. Chartrand, S. Degterov, G. Eriksson, K. Hack, R. Mahfoud, J. Melancon, A. Pelton and S. Peterson, "FactSage Thermochemical Software and Databases", *Calphad*, 26 (2) (2002), 189-228.

[57] A. Kyllo and G. Richards, "A Kinetic Model of the Peirce-Smith Converter: Part I. Model Formulation and Validation", *Metallurgical Transactions B*, 29 (1) (1998), 239-249.

[58] J. Moore, C. Stanitski and P. Jurs, "Gas Density and Molar Masses", Section 10.6 of *Principles of Chemistry: The Molecular Science*, Brooks/Cole (2010), 359-360.

[59] P. Liley, G. Thomson, T. Daubert and E. Buck, "Physical and Chemical Data", Part 2 of *Perry's Chemical Engineers' Handbook*, McGraw-Hill (1997).

[60] H. Lee, "Heat of Formation, standard entropies and heat capacities", Appendix 1 of *Chemical Thermodynamics for Metals and Materials Engineers*, Imperical College Press (1999), 275-281.

[61] R. Bird, W. Stewart and E. Lightfoot, "The Equations of Change for Nonisothermal Systems", Chapter 11 of *Transport Phenomena*, John Wiley and Sons (2002), 333-373.

[62] H. Lee, "Fundamental Principles and Functions", Chapter 1 of *Chemical Thermodynamics for Metals and Materials Engineers*, Imperical College Press (1999), 1-60.

[63] J. Stewart, "The Fundamental Theorem Of Calculus", Section 5.4 of *Calculus: Concepts and Contexts*, Cengage Learning (2009), 367-374.

[64] D. Lynden-Bell and R. Lynden-Bell, "On the negative specific heat paradox", *Monthly Notices of the Royal Astronomical Society*, 181 (1977), 405-419.

[65] W. Cheney, "Implicit Function Theorems", Section 3.4 of *Analysis for Applied Mathematics*, Springer-Verlag (2001), 312-315.

[66] A. Navarra and J. Kapusta, "Decision-making Software Development for the Incremental Improvement of Peirce-Smith Converters", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 231-250.

[67] J. Dottorro, "Overview", Chapter 1 *Convex Optimization and Euclidean Distance Geometry*, Meboo Publishing (2005), 19-32.

[68] J. Nocedal and S. Wright, "Quadratic Programming", Chapter 16 of *Numerical Optimization*, Springer (2006), 449-496.

[69] B. Liengme, "Solver", Chapter 12 of *Guide to Microsoft Excel 2007 for Scientists and Engineers*, Elsevier (2009), 211-229.

[70] "Qosmio©gaming and creativity unleashed", Toshiba website, *us.toshiba.com/computers/laptops/qosmio/*, accessed November 24th 2012.

[71] D. Gay, *IBM ILOG CPLEX Optimization Studio, Getting Started with CPLEX*, IBM white paper, Version 12, Release 4 (2011).

[72] R. Fourer, D. Gay and B. Kernighan, "Interaction with Solvers", Chapter 14 of *AMPL, A Modeling Language for Mathematical Programming*, Thompson Learning (2002), 275-317.

[73] H. Noguchi, J. Tani, Y. Shimai, H. Kawaguchi and M. Yoshimoto, "Parallel-Processing VLSI Architecture for Mixed Integer Linear Programming", *International Symposium on Circuits and Systems*, (2010), 2362-2365.

[74] T. Cormen, C. Leiserson, R. Rivest and C. Stein, "Greedy Algorithms", Chapter 16 of *Introduction to Algorithms*, McGraw-Hill (2001), 370-404.

[75] M. Ek and P. Olsson, "Recent Developments on the Peirce-Smith Converting Process at the Rönnskär Smelter", *Converter and Fire Refining Practices*, (The Minerals, Metals and Materials Society Conference, 2005), 19-26.

[76] M. Ek and P. Olsson, "Converting and Casting at Bolidens Rönnskär Smelter 2009 An Update", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 127-132.

[77] Search result for "AMPL", The Free Dictionary by Farlex, *acronyms.thefreedictionary.com/AMPL*, accessed November 24th 2012.

[78] Y. Shafranovich, "Common Format and MIME Type for Comma-Separated Values (CSV) Files", *The Internet Society*, RFC 4180 (2005).

[79] MOSEK ApS, "MOSEK and AMPL", Chapter 5 of *The MOSEK Optimization Tools Manual*, Version 6, Release 148 (2012), 21-28.

[80] K. Holmström, A. Göran and M. Edvall, "TOMLAB/MINLP Solver Reference", Chapter 3 of *User's Guide for TOMLAB/MINLP*, TOMLAB Optimization (2007), 6-41.

[81] H. Dixon, "What's New in Excel 2007?", Chapter 1 of *Excel 2007: Beyond the Manual*, Apress (2007), 1-24

[82] J. Mueller, "VBA Programming in Excel", Chapter 14 of *VBA for Dummies*, Wiley Publishing (2007), 305-327

[83] B. Salt and E. Cerilli, "Evolution of the Converter Aisle at Xstrata Nickel's Sudbury Smelter", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 135-149.

[84] "Ferronickel Pellet", *Material Safety Data Sheet*, Vale-Inco (2010).

[85] W. Gasior, Z. Moser and A Debski, "Heat of formation of $FeNi_{70}$, $FeNi_{73.5}$ and $FeNi_{80}$ ordered alloys from the homogenous region of the $FeNi_3$ phase", *Journal of Alloys and Compounds*, 487 (2009), 132-137.

[86] L. Rolla, "The Heat Capacity of Alloys, Amalgams and Intermetallic Compounds", *International Critical Tables of Numerical Data: Physics, Chemistry and Technology*, 5 (1929), 118-122.

[87] W. Cheney, "Measure and Integration", Chapter 8 of *Analysis for Applied Mathematics*, Springer-Verlag (2001), 381-427.

[88] R. Bird, W. Stewart and E. Lightfoot, "Energy Transport by Radiation", Chapter 16 of *Transport Phenomena*, John Wiley and Sons (2002), 487-509.

[89] W. Cheney, "Distributions", Chapter 5 of *Analysis for Applied Mathematics*, Springer-Verlag (2001), 246-286.

[90] R. Bird, W. Stewart and E. Lightfoot, "Shell Energy Balances and Temperature Distributions in Solids and Laminar Flow", Chapter 10 of *Transport Phenomena*, John Wiley and Sons (2002), 290-332.

[91] R. Bird, W. Stewart and E. Lightfoot, "Interphase Transport in Nonisothermal Systems", Chapter 14 of *Transport Phenomena*, John Wiley and Sons (2002), 422-453.

[92] S. Le Digabel, "NOMAD: Nonlinear Optimization with the MADS Algorithm", *ACM Transactions on Mathematical Software*, 37(43) (2011), Article 44.

[93] I. Harjunkoski and I. Grossmann, "Decomposition techniques for multistage scheduling problems using mixed integer and constraint programming methods", *Computers and Chemical Engineering*, 26 (2002), 1533-1552.

[94] R. Hummel, "The First Materials (Stone Age and Copper-Stone Age)", Chapter 1 of *The Substance of Civilization*, Springer (2004), 3-11.

[95] J. Douglas, "Treatment of Copper Mattes in the Bessemer Converter", *Transactions of the American Institute of Mining and Metallurgy*, 8 (1899), 2-48.

[96] W. Alexander, "A Brief Review of the Development of the Copper, Zinc and Brass Industries in Great Britain from AD 1500 to 1900", *Murex Review*, 1 (15) (1955).

[97] G. Lalev, J. Lim, N. Munirathnam, G. Choi, M Uchikoshi, K. Mimura and M. Isshiki, "Concentration Behavior of Non-Metallic Impurities in Cu Rods Refined by Argon and Hydrogen Plasma-Arc Zone Melting", *Materials Transactions*, 50 (3) (2009), 618-621.

[98] H. De la Beche, "Mining, Quarrying and Metallurgical Processes and Products, Great Exhibition", *Lectures on the Results of the Great Exhibition of 1851*, 37-73.

[99] A. Pelletier, P. Mackey, L. Southwick and A. Wraith, "Before Peirce and Smith - The Manhes Converter: Its Development and Some Reflections for Today", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 29-49.

[100] M. Marinigh, "Technology and Operational Improvements in Tuyère Punching, Silencing, Pyrometry and Refractory Drilling Equiment", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 199-215.

[101] C. Ramirez, P. Ruz, G. Riveros, A. Warczok and R. Treimer, "Chrome-Magnesite Refractory Corrosion with Olivine Slag of High Cuprous Oxide Content", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 71-80.

[102] G. Oprea, W. Lo, T. Troczynski and J. Rigby, "Corrosion of Refractories in Peirce Smith Converters", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 83-99.

[103] A. Rigby, "Controlling the processing parameters affecting the refractory requirements for Peirce-Smith converters and anode refining vessels", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 213-222.

[104] T. Lehner and C. Samuelsson, "Converting and Refining - Experiences in Ferrous and Non-Ferrous Metallurgy", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 53-69.

[105] R. Van Schalkwyk, J. Eksteen, J. Petersen, E. Thyse and G. Akdogan, "An experimental evaluation of the leaching kinetics of PGM-containing Ni-Cu-Fe-S Peirce Smith converter matte, under amospheric leach conditions", *Minerals Engineering*, 24 (6), 524-534.

[106] E. Thyse, G. Akdogan and J. Eksteen, "The effect of changes in iron-endpoint during Peirce-Smith converting on PGE-containing nickel converter matte mineralization", *Minerals Engineering*, 24 (7), 688-697.

[107] W. Wendt, M. Alden, B. Bjorkman, T. Lehner and W. Persson, "Controlling copper conversion via optical spectroscopy", *Journal of Metals*, 39 (10) (1987), 14-17.

[108] E. Dudgeon, "Measurement of sulphur dioxide content of copper converter gases as aid to converter control", *Canadian Institute of Mining and Metallurgy - Mineral Society of Nova Scotia*, 69 (1966), 430-437.

[109] L. Larson, "General improvement in converter practice", *Mining Congress Journal*, 13 (9) (1927), 727-728.

[110] T. Maruyama, T. Saito and M. Kato, "Improvements of converter's operation at Tamano Smelter", *Sulfide Smelting* (The Minerals, Metals and Materials Society Conference, 1998), 219-227.

[111] S. Tanaka, M. Hamamoto, M. Hashimoto and S. Udo, "Operation and improvements on Peirce-Smith converters at the Tamano Smelter", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 79-88.

[112] M. Coleman and G. Money, "Increasing Capacity and Productivity in the Metals Markets through Pneumatic Conveying and Process Injection Technologies", (The Minerals, Metals and Materials Society Conference, 2009), 217-230.

[113] O. Rojas and J. Sanhueza, "Hernan Videla Lira copper smelter modernization", *Pyrometallurgy of Copper* (Copper 99 - Cobre 99 International Conference, 1999), 17-27.

[114] O. Pasca, J. Bryant, P. Safe and B. Wiggins, "Peirce-Smith converter hood improvements at BHP copper", *Pyrometallurgy of Copper* (Copper 99 - Cobre 99 International Conference, 1999), 149-159.

[115] P. Safe, J. Deakin and S. Matson, "Effective design of converter hoods", *Sulfide Smelting* (The Minerals, Metals and Materials Society Conference, 2002), 99-110.

[116] G. Achurra, P. Chacana, J. Büchi and F. Condore, "Present and future of caletones", *Yazawa International Symposium* (The Minerals, Metals and Materials Society Conference, 2003), 483-494.

[117] K. Mori, N. Nagai, K. Morita and O. Nakano, "Recent Operation and Improvement at the Sumitomo Toyo Peirce-Smith Converters", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 151-160.

[118] W. Drummond and J. Deakin, "Water-cooled hood system for Peirce-Smith converters and similar furnace vessels", *Journal of Metals*, 51 (5) (1999), 40-41.

[119] I. Bauer and H. Velten, "Reduction of fugitive emissions in the secondary smelter of Norddeutsche Affinerie AG", *World of Metallurgy - ERZMETALL*, 59 (2) (2006), 74-80.

[120] L. Contreras, P. Reyes, B. Martinich and R. Bustamante, "Gas and cleaning at the Potrerillos smelter", *Pyrometallurgy of Copper* (Copper 99 - Cobre 99 International Conference, 1999), 119-132.

[121] W. Davenport, "Copper smelting to the year 2000", *Canadian Institute of Mining, Metallurgy and Petroleum Bulletin*, 73 (813) (1980), 152-158.

[122] P. Chaubal, "Effect of oxygen enrichment on elimination of As, Sb, Bi during converting", *Transactions of the Institute of Mining and Metallurgy, Section C: Mineral Processing and Extractive Metallurgy*, 98 (1989), 83-84.

[123] A. Bustos, J. Kapusta, B. Macnamara and M. Coffin, "High oxygen shrouded injection at Falconbridge", *Pyrometallurgy of Copper* (Copper 99 - Cobre 99 International Conference, 1999), 93-106.

[124] J. Brimacombe, S. Meredith and R. Lee, "High-Pressure Injection of air into a Peirce-Smith Copper Converter", *Metallurgical Transctions B*, 15 (2) (1984), 243-250.

[125] J. Kapusta, N. Wachgama and R. Pagador, "Implementation of Air Liquide Shrouded Injector (ALSI) Technology at the Thai Copper Industries Smelter", *Pyrometallurgy of Copper* (Copper 2007 - Cobre 2007 International Conference, 2007), 484-500.

[126] R. Pagador, N. Wachgama, C. Khuankla and J. Kapusta, "Operation of the Air Liquide Shrouded Injector (ALSI) Technology in a Hoboken Siphon Converter", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 367-381.

[127] J. Leroux, B. Langlois, Y. Massé, X. Guo and P. Mackey, "Development of new bath smelting technology at Mines Gaspé", *Pyrometallurgy of Copper* (Copper 99 - Cobre 99 International Conference, 1999), 399-416.

[128] T. Kawai, M. Nishiwaki and S. Hayashi, "Copper concentrate smelting in Peirce-Smith converters at Onahama Smelter", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 119-123.

[129] F. Longworth, "Smelting copper concentrates in a converter", *Mining and Metallurgy*, 5 (214) (1924), 485-486.

[130] M. Boisvert, G. Janneteau, J. Landry, C. Levac, D. Perron, F. McGlynn, M. Zamalloa and F. Porretta, "Design and construction of the Noranda Converter at the Horne

Smelter", *Sulfide Smelting* (The Minerals, Metals and Materials Society Conference, 1998), 569-583.

[131] C. Caballero, A. Moyano, P. Morales, C. Toro, H. Jara, L. Guzmán and R. Díaz, "A Dynamic Simulation for the Validation Tests of the Codelco-Chile Continuous Converting Process", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 263-272.

[132] A. Moyano, C. Caballero, C. Toro, P. Morales and J. Font, "The Validation of the Codelco-Chile Continuous Converting Process", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 349-360.

[133] C. Chen, J. Zhang, M. Bai and S. Wei, "Investigation on the copper content of matte smelting slag in Peirce-Smith converter", *Journal of University of Science and Technology Beijing: Mineral Metallurgy Materials* (English Edition), 8 (3) (2001), 177-181.

[134] P. Tan and P. Vix, "Modelling of slag blow in copper Peirce-Smith Converters", *First Extractive Metallurgy Operators' Conference* (2005), 109-115.

[135] P. Tan and P. Vix, "Control of magnetite formation during slag-making in copper P-S converter", *Converter and Fire Refining Practices* (The Minerals, Metals and Materials Society Conference, 2005), 247-258.

[136] N. Cardona, P. Mackey, P. Coursol, R. Parada and R. Parra, "Optimizing Peirce-Smith Converters Using Thermodynamic Modeling and Plant Sampling", *Journal of Metals*, 64 (5) (2012), 546-550.

[137] N. Amlnlzadeh and S. Mansouri, "Thermo-chemical model of the Pierce-Smith copper converter", *Journal of Process Mechanical Engineering*, 221 (3) (2007), 129-138.

[138] S. Marcuson, S. Ellor and C. Diaz, "Heat and mass balances in copper and nickel smelting process", *Kellogg International Symposium on Quantitative Description of Metal Extraction Processes* (The Minerals, Metals and Materials Society Conference, 1991), 179-193.

[139] H. Kellogg, "Thermochemistry of Nickel-Matte Converting", *Keynote Address* (Conference of Metallurgists, 1986), 95-128.

[140] A. Bustos, S. Ip, G. O'Connell, G. Kaiura and J. Toguri, "Converting Simulation at Falconbridge Limited", *Extractive Metallurgy of Nickel and Cobalt* (The Metallurgical Society Conference, 1988), 179-193.

[141] A. Kyllo and G. Richards, "A Kinetic Model of the Peirce-Smith Converter: Part II. Model Application and Discussion", *Metallurgical Transactions B*, 29 (1) (1998), 251-259.

[142] P. Tan, "Applications of Thermo-Chemical and Thermo-Physical Modeling in the Copper Converter Industries", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 273-295.

[143] A. Alyaser and J. Brimacombe, "Oxidation kinetics of molten copper sulfide" , *Metallurgical Transactions B*, 26 (1) (1995), 25-40.

[144] F. Carrillo, R. Hernández, J. Martinez and A. Roselló, "Kinetics of the copper blow in the Peirce-Smith converter", *Información Tecnológica*, 15 (5) (2004), 33-36.

[145] A. Cervantes, C. Real, M. Palomar, L. Hoyos, M. Gutierrez and J. Gonzalez, "Minimum Numerical Model of a Peirce-Smith Converter", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 297-309.

[146] D. Chibwe, G. Akdogan, C. Aldrich and P. Taskinen, "Characterisation of phase distribution in a Peirce-Smith converter using water model experiments and numerical simulation", *Transactions of the Institutionsof Mining and Metallurgy*, 120 (3) (2011), 162-171.

[147] D. Chibwe, G. Akdogan, C. Aldrich and R. Eric, "CFD Modelling of Global Mixing Parameters in a Peirce-Smith Converter with Comparison to Physical Modelling", *Chemical Product and Process Modeling*, 6 (1) (2011), 1-30.

[148] N. Papadokos, "Integrated airline scheduling", *Computers and Operations Research*, 36 (2009), 176-195.

[149] M. Pinedo, "Deterministic Models: Preliminaries", Chapter 2 of *Scheduling: Theory, Algorithms and Systems*, Springer (2012), 13-34.

[150] G. Dantzig, "Programming of Interdependent Activities: II Mathematical Model", *Econometrica*, 17 (3) (1949), 200-211.

[151] L. Tang, J. Liu, A. Rong and Z. Yang, "A review of planning and scheduling systems and methods for integrated steel production", *European Journal of Operational Research*, 133 (2001), 1-20.

[152] A. Ghosh and A. Chatterjee, "Overview of Modern Steelmaking", Chapter 3 of *Ironmaking and Steelmaking: Theory and Practice*, Phi Learning (2011), 38-89.

[153] C. Morris and S. Wallden, "Development of the Kaldo furnace smelting technique and its application for top blown rotary converter (TBRC) copper smelting and refining", *Transactions of the Metallurgical Society of the American Institute of Mining, Metallurgical and Petroleum Engineers*, (1976), 426-438.

[154] J. Gonzalez, C. Real, M. Palomar, L. Hoyos and M. Gutierrez, "CFD simulation of a copper converter with bottom air injection", *Extraction and Processing*, (The Minerals, Metals and Materials Society Conference, 2008), 323-333.

[155] J. Gonzalez, C. Real, M. Palomar, L. Hoyos, M. Gutierrez and R. Miranda, "CFD simulation gas-liquid flow in a copper converter with bottom air injection", *International Journal of Chemical Reactor Engineering*, 6 (1) (2008).

[156] W. Davenport, M. King, M. Schlesinger and A. Biswas, "Mitsubishi Continuous Smelting/Converting", Chapter 13 of *Extractive Metallurgy of Copper*, Pergamon (2002), 199-216.

[157] J. Wood, R. Matusewicz and M. Reuter, "Ausmelt C3 Converting", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 397-406.

[158] S. Nikolic, J. Edwards, A. Burrows and G. Alvear, "ISACONVERT - TSL Continuous Copper Converting Update", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 407-414.

[159] I. Kojo, M. Lahtinen and E. Miettinen, "Flash Converting - Sustainable Technology Now and in the Future", *International Peirce-Smith Converting Centennial* (The Minerals, Metals and Materials Society Conference, 2009), 383-395.

[160] V. Dale-Jones, "Mining town moves to avoid pollution", BBC News, Novemeber 4th, 2002.

[161] T. Mäkinen and P. Taskinen, "State of the art in nickel smelting: direct Outokumpu nickel technology", *Mineral Processing and Extractive Metallurgy* , 117 (2) (2008), 86-94.

[162] R. Fourer, D. Gay and B. Kernighan, "Introduction", *AMPL, A Modeling Language for Mathematical Programming*, Thompson Learning (2002), xv-xxi.

[163] V. Chvátal, "General LP Problems: Solutions by the Simplex Method", Chapter 8 of *Linear Programming*, W. H. Freeman and Company (1983), 118-136.

[164] T. Cormen, C. Leiserson, R. Rivest and C. Stein, "Linear Programming", Chapter 27 of *Introduction to Algorithms*, McGraw-Hill (2001), 770-821.

[165] G. Schay, "Systems of Linear Equations, Matrices", Chapter 2 of *A Concise Introduction to Linear Algebra*, Springer (2010), 41-98.

[166] G. Dantzig, "Maximization of a linear function of variables subject to linear inequalities", Chapter 21 of *Activity Analysis of Production and Allocation*, John Wiley and Sons (1951), 339-347.

[167] R. Cottle, E. Johnson and R. Wets, "George B. Dantzig, 1914-2005", *Notices of the American Mathematics Society*, 54 (3) (2007), 344-362.

[168] V. Chvátal, "The Revised Simplex Method", Chapter 7 of *Linear Programming*, W. H. Freeman and Company (1983), 97-117.

[169] J. Nocedal and S. Wright, "Linear Programming: The Simplex Method", Chapter 13 of *Numerical Optimization*, Springer (2006), 355-391.

[170] F. Hillier and G. Lieberman, "Solving Linear Programming Problems: The Simplex Method", Chapter 4 of *Introduction to Operations Research*, McGraw-Hill (2010), 89-160.

[171] R. Bronson and G. Naadimuthu, "Linear Programming: The Simplex and the Dual Simplex Methods", Chapter 3 of *Operations Research*, Schaum's Outline Series, McGraw-Hill (1997), 32-55.

[172] V. Klee and G. Minty, "How good is the simplex algorithm?", *Inequalities*, III (1972), 159-175.

[173] V. Chvátal, "The Ellipsoid Method", Appendix to *Linear Programming*, W. H. Freeman and Company (1983), 443-454.

[174] J. Nocedal and S. Wright, "Linear Programming: Interior-Point Methods", Chapter 14 of *Numerical Optimization*, Springer (2006), 392-420.

[175] N. Karmarkar, "A New Polynomial-Time Algorithm For Linear Programming", *Combinatorica*, 4 (4) (1984), 373-395.

[176] R. Bronson and G. Naadimuthu, "Linear Programming: Extensions", Chapter 5 of *Operations Research*, Schaum's Outline Series, McGraw-Hill (1997), 87-123.

[177] J. Gondzio, "Interior point methods 25 years later", *European Journal of Operational Research*, 218 (2012), 587-601.

[178] P. Gill, W. Murray, M. Saunders and M. Wright, "Sparse Matrix Methods in Optimization", *Journal on Scientific and Statistical Computing*, 5 (3) (1984), 562-589.

[179] E. Yildirim and S. Wright, "Warm-Start Strategies in Interior-Point Methods for Linear Programming", *Journal on Optimization*, 12 (3) (1984), 782-810.

[180] V. Chvátal, "The Duality Theorem", Chapter 5 of *Linear Programming*, W. H. Freeman and Company (1983), 54-70.

[181] S. Thornton and J. Marion, "Hamilton's Principle, Lagrangian and Hamiltonian Dynamics", Chapter 7 of *Classical Dynamics*, Thomson (2004), 228-286.

[182] V. Chvátal, "Matrix Games", Chapter 15 of *Linear Programming*, W. H. Freeman and Company (1983), 228-239.

[183] J. Nocedal and S. Wright, "Interior-Point Methods for Nonlinear Programming", Chapter 19 of *Numerical Optimization*, Springer (2006), 563-597.

[184] F. Hillier and G. Lieberman, "The Dual Simplex Method", Section 7.1 of *Introduction to Operations Research*, McGraw-Hill (2010), 276-280.

[185] F. Hillier and G. Lieberman, "Integer Programming", Chapter 11 of *Introduction to Operations Research*, McGraw-Hill (2010), 464-536.

[186] A. Land and A. Doig, "An Automatic Method of Solving Discrete Programming Problems", *Econometrica*, 28 (3) (1960), 497-520.

[187] T. Achterberg, T. Koch and A. Martin, "Branching rules revisited", *Operations Research Letters*, 33 (2005), 42-54.

[188] R. Gomory, "Outline of an algorithm for integer solutions to linear programs", *Bulletin of the American Mathematical Society*, 5 (1958), 275-278.

[189] V. Chvátal, "Edmonds polytopes and a hierarchy of combinatorial problems", *Discrete Mathematics*, 4 (4) (1973), 305-337.

[190] P. Ow and T. Morton, "Filtered beam search in schduling", *International Journal of Production Research*, 26 (1) (1988), 35-62.

[191] M. Pinedo, "More Advanced General Purpose Procedures", Chapter 15 of *Scheduling: Theory, Aplications and Systems*, Springer (2012), 399-429.

[192] M. Pinedo, "Mathematical Programming: Formulations and Applications", Appendix A of *Scheduling: Theory, Aplications and Systems*, Springer (2012), 559-571.

[193] G. Owen, "Cutting planes for programs with disjunctive constraints", *Journal of Optimization Theory and Applications*, 11 (1) (1973), 49-55.

[194] I. Grossmann, "Advances in mathematical programming models for enterprise-wide optimization", *Computers and Chemical Engineering*, 47 (2012), 2-18.

[195] O. Boni, F. Fournier, N. Mashkif, Y. Naveh, A. Sela, U. Shani, Z. Lando and A. Modai, "Applying Constraint Programming to Incorporate Engineering Methodologies into the Design Process of Complex Systems", *Innovative Applications of Artificial Intellegence Conference* (2012), 2262-2268.

[196] E. Bajalinov, "Introduction to LFP", Chapter 3 *Linear Fractional Programming: Theory, Methods, Applications and Software* (2003), 41-74.

[197] F. You, P. Castro and I. Grossmann, "Dinkelbach's algorithm as an efficient method to solve a class of MINLP models for large-scale cyclic scheduling problems", *Computers and Chemical Engineering*, 33 (2009), 1879-1889.

[198] C. Wen and H. Wu, "Using the Dinkelbach-type algorithm to solve the continuous-time linear fractional programming problems", *Journal of Global Optimization*, 49 (2011), 237-263.

APPENDIX A

LITERATURE REVIEW OF PEIRCE-SMITH CONVERTING

A.1 Origins of PS Converting

Advances in metallurgy are closely related to advances in civilization. Copper toolmaking marked the end of the Stone Age [94], and the copper-based electrical system is a hallmark of modern society. Superalloys employ nickel, copper, iron and numerous other metals to elicit the special properties that are fundamental to modern technology.

Early industrial copper production was centered in the copper smelters of Swansea, Wales [29]. They employed the so-called Welsh process, which had been guarded by a "ridiculous secrecy" [29, 95]. Through some efforts, industrial historians were eventually able to collect documentation and testimony describing this archaic process. According to Alexander [96], the ore at Swansea was broken up into small lumps, and was hand-sorted by girls; the selected ore underwent at least seven alternations between roasting and smelting. The reversion from smelting back to roasting implies a deactivation (re-solidification), which expels impurities through microstructural segregation, thus a 35% copper ore is converted into a 98% blister copper; this is comparable to the modern practice of zone-melting, which is used to convert 99.99% copper into 99.9995% copper [97]. Interestingly, the Welsh blister was subject to a fire refining stage that was quasi-modern, employing greenwood as the reductant hydrocarbon [15, 96], rather than the modern alternatives: natural gas, butane, diesel, etc.

In 1856, Bessemer's innovation in steelmaking featured a clear division between smelting and converting, which had not been present in the secretive Welsh process for coppermaking [29, 95, 96]. The division was first-of-all conceptual, that (1) roasting, (2) smelting and (3) converting should be separate, and followed by (4) refining. This conceptual change may already have been discussed in the Great Exhibition of 1851 [98], which renounced the secretive Welsh culture, in favour of scientific exchange under intellectual protection. The scientific openness, and the notion of a "coppermaking community", continues to be a valuable support for technological development.

Following the Great Exhibition and Bessemer's pneumatic steelmaking furnace, there was a tremendous effort to overtake the Welsh process. In the 1870's, Hollway published experimental results for the pneumatic conversion of copper matte, which later contributed

to the work of Baggaley, described below. The first successful pneumatic copper converting process was developed by Manhès [29, 99] in 1878, in vertically orientated furnaces, similar to that of the Bessemer; this approach used two separate furnaces: one for the Slag-Blow stage, and another for the Copper-Blow stage. In 1885, Schumacher was the first to combine both stages within a single vertical furnace, known as the Parrot converter, which slightly predates the Great Falls converter [29]. In 1890, Stallman developed a trough shaped converter with a square cross-section [29].

Following his initial successes, Manhès worked with David to create the first horizontal converter [29, 99], introduced in Leghorn, Italy, in 1891. Shortly after, the Copper Queen Smelter tested a similar unit. Thus the Manhès-David converter has been referred to as the Leghorn converter, as well as the Copper Queen converter [29]. This converter had a lower height than its vertical counterpart, and was thus noted for its ease of manipulation.

The Manhès-David converter has the appearance of a Peirce-Smith converter. However, all converter linings were being made of acid refractories, i.e. silica [29, 99]. Of course, this lining was actively melded with the iron that was in the matte, thus forming slag. These acid linings were consumed extremely quickly, and would barely endure a single day of operation [29]. Typical smelter operations would have three different states of repair for the converters, which were evenly split: relining, curing and operational.

Hixon understood that the acid lining was being consumed as if it were a reagent, so in 1892 he attempted to line a Parrot converter with magnesite instead [29]. This basic lining endured the chemical attack from the bath, but did not provide sufficient thermal insulation. Nonetheless, Baggaley revisited the notion of magnesite lining, and implemented a water-cooling jacket that would cool the steel shell. Baggaley successfully ran a campaign in horizontal converter, from October 7, 1905 to January 31, 1906, without having to change the lining, hence validating the water-cooled basic lining. The Baggaley converter is considered to be the immediate predecessor of the Peirce-Smith converter [29].

Baggaley had been an avid researcher, learning from the successes and failures of Hollway and Hixon, and adapting them to the Manhès-David geometry [29]. He was able to bring these ideas to fruition partly because of his close relationship to the Westinghouse interest; funding was not an issue. However, he retired from his work after his son had tragically died in a fire on February 12, 1906 [29]. Later in 1906, representatives of the Guggenheim interest met with Baggaley and offered him a chance to modify their sixteen smelters and refineries, but Baggaley turned them down. Peirce and Smith were also present at this meeting, and eventually continued the work of Baggaley [29].

Peirce and Smith made four notable changes to Baggaley's design [29, 30]. Firstly, they installed thicker refractories in the tuyere belt. Secondly, they installed expansion joints to allow for the thermal expansion of the lining. Thirdly, they introduced pouring spouts on the side of the converter. Lastly, they incorporated replaceable tuyeres. The first experimental unit was installed in 1909, and the first industrial unit was installed in 1910.

Following 1910, the Peirce-Smith converter became pervasive around the world, as the Welsh had clearly lost their stronghold. Peirce-Smith converters have remained dominant ever since, with the two enduring qualities of simplicity and adaptability.

A.2 Incremental Improvements in PS Converting

Over the last century, considerable improvements have been made to the original Peirce-Smith Converter. These vessels continue to be included in new plant designs [40, 42], and are not likely to be forgotten in the near future.

The list of improvements is extensive [42, 100]: tuyere silencers, tuyere punching, bustle main, refractory materials, mouth and hood design, electronic controls, etc. Mechanized tuyere punching has decreased inter-cycle time. Better refractories, bricking practice, and thermochemical controls have all extended the life of the refractory lining. Use of oxygen enrichment has assisted in the heat balance, also bringing improved productivity [42].

Even before the usage of basic lining, there was a recognized link between refractory wear and slag chemistry. This has remained a concern even with modern chrome-magnesite lining [32, 102]. It is generally desirable to avoid needlessly long exposure to the corrosive copper oxidic slag [103, 104], yet an underoxidized blister is demanding on the refining furnaces [104]. Slag chemistry has also been related the recovery of platinum group metals from nickel-bearing mattes [105, 106]. This supports the implementation of end-point controls, to decide exactly when to cease the blowing action [35, 36, 107]. Early end-point controls measured changes in the rate of SO_2 production [108], but the more modern Semtech technology uses the optical detection of lead compounds in the fume dust [107].

There has been an ongoing drive toward longer Peirce-Smith converter that, interestingly, has been documented since the early usage [109]. Indeed, a larger batch capacity implies a larger bath volume, but the need for effective air injection has limited the diameter. Therefore, the increased volume has been manifested mainly as a length increase. Nonetheless, the converter length has been limited by the conveying systems for flux and secondary feeds. There have been several documented cases in which productivity increases have resulted from

the combined effect of longer converters and enhanced fluxing and secondary feeding systems [110, 111, 112].

Since the late 1990's, there have been numerous reports of smelter upgrade projects that emphasize offgas handing [113, 114, 115, 116, 117], which often coincide with the retirement of an old-fashioned reverberatory furnace [113, 116]. Water-cooled hoods are able maintain a tighter closure over the mouth of Peirce-Smith converters, diminishing the fugitive emissions, while maintaining a favorable gas strength and temperature [118]. It has become common to encapsulate a water-cooled hood within a secondary hood, to capture any gasses that escape the first hood [42, 115, 117, 119]. It has also become standard practice to clean the gases with electrostatic precipitators and/or wet scrubbers [42, 120].

In 1980, Davenport had foreseen the trend toward stricter environmental regulations, and expected that Hoboken converters would become more dominant [121]; but rather than the bulky siphon construction of Hoboken, the smelting community has opted for the slightly less bulky construction secondary hoods [42]. In any case, the environmental regulations have had a direct impact that has been well received. There is now a friendly rivalry between smelters, to capture as high a percentage of SO_2 as possible, and transform it into acid. Roughly speaking, a capture of above 95% is considered to be "boast-worthy".

Oxygen enrichment has been used in copper and nickel production since the 1950's, but is normally limited to below 30% [44], to maintain the integrity of the refractory lining. Oxygen enrichment has had a major impact to control the heat balance of Peirce-Smith converters [42], and is linked to the management of arsenic, antimony and bismuth [122]. In 1995, Bustos et al. successfully applied the concept of high-pressure shrouded injection into the non-ferrous industry [44], which permitted oxygen enrichment up to 60%; this approach had already been successful for steelmaking. Bustos et al. employed an experimental Hoboken unit on a complex copper-lead-iron matte. Within a few years, this technology was commercialized as Air Liquid Shrouded Injection (ALSI), and was installed at the Falconbridge Smelter in Sudbury, in the so-called Slag Make Converter [44, 45, 46]. The Slag Make Converter is an enhanced Peirce-Smith converter that removes most of the iron and associated sulfur from a nickel-bearing matte; the product is then sent to conventional Peirce-Smith converters, which act as finishing vessels [46].

The Falconbridge Smelter has been a showcase for the application of the ALSI technology in the nickel industry, which not only allows higher oxygen enrichment, but also avoids the need for tuyere punching [45]. At pressures above 250 kpa, the bubble-forming regime gives way to jetting. The jetting regime does not lead to the confining accretions that would

require punching. This seems to be an underutilized concept that could be adapted even to traditional tuyeres [124]; indeed, a higher price for gas compression could be offset by the price of tuyere and refractory maintenance [66].

In 2006, the Thai Copper Smelter became the first copper smelter to employ the ALSI technology. This was a new smelter that had undergone a troubled beginning, marred by the Asian economic crisis of the late 1990's, and the devaluing of the Thai currency. The plant had operated for a year in 2004 until there was a catastrophic industrial accident [125]. Finally in 2006, the smelter was reopened, after having installed a number of technological upgrades; the converter aisle featured Hoboken converters with the ALSI. The Thai smelter ran successfully for roughly two years [126], and it was hoped that other copper installations would also employ the ALSI. However, the success of Thai Copper was short-lived, as it was closed during the economic crash of 2008. At the International Peirce-Smith Converting Centennial Symposium of 2009, the representatives from Thai Copper had impressive results to share, but the main author could not be present, as the smelter had already ended its operations.

An apparent difficulty with the ALSI is the need for cold charges, such as reverts and scraps. Some have speculated that the excess heat could be balanced by feeding moist concentrate. This concentrate would therefore bypass the drying/roasting stages, as well as the smelting stage. The feeding of Peirce-Smith converter with concentrate would not be a new idea [127, 128]. There are reports of concentrate feeding as early as the 1920's [129]. The Noranda Reactor and Teniente Converter are smelting furnaces that borrowed concepts from Peirce-Smith Converting, including horizontal geometry, basic refractory lining, and tuyere arrangement [8]. There appear to be only superficial differences between the Noranda Reactor and the Teniente Converter, with regard to the charging and discharging mechanisms. In this case, the term "Teniente Converter" is a misnomer since this vessel is more accurately described as a smelting furnace rather than a converter, as it is fed mainly with concentrate.

The Noranda Reactor is not to be confused with the Noranda Converter [12]. Firstly, the Noranda Reactor was originally designed to continuously produce blister copper, featuring three coexisting liquid phases: slag, matte and blister copper. The blister copper could be continuously drained from the bottom of the melt, as the slag is drained from the top. In the modern usage of the Noranda Reactor, however, the continuous product is white metal rather than blister [130]; the original three-phase concept was found to exhibit hazardous arsenic transport, which caused a poisonous gas release during the electrorefining stage. Nonetheless, the main concepts of the Noranda Reactor have been incorporated into the

Noranda Converter, which continuously converts white metal into blister [12].

The evolution from the Noranda Reactor to the Noranda Converter is being mimicked, as experiments are being run on the Teniente Converter [131, 132]. Indeed, the so-called Continuous Codelco Converting (CCC) uses a Teniente Converter, but the feed is a blend of molten and solidified matte, and the slag has an olivine chemistry $(Ca,Fe)SiO_4$ instead of fayalite Fe_2SiO_4. The CCC process builds on some of the positive attributes of Noranda Reactor/Converter, but has not been implemented on a commercial scale. Other alternatives to Peirce-Smith Converting are presented in Appendix A.5.

There will undoubtedly be more incremental improvements in Peirce-Smith Converting, to accommodate future feeds. Given the advances in computational modeling, the acceptance of these improvements will be supported by simulations.

A.3 Computational Modeling of PS Converting

The pyrometallurgical community has long recognized two main branches of Peirce-Smith Converting Modeling: chemical thermodynamics and chemical kinetics. In recent years, the theme of kinetics has been expanded to include mixing and computational fluid mechanics (CFD).

Thermodynamic modeling has generally increased in complexity more realistic simulations, especially with regards to slag chemistry [133, 134, 135, 136]. However, Alminizadeh preferred a simplified model in 2007 [137]. This approach lacks realism, particularly in the representation of magnetite within the slag. While this work is not particularly well suited for simulation, it seems to be appropriate for optimization; it may be compared to the work of Navarra et al. [53], which avoids the use of Gibbs free energy surfaces and phase diagrams, by adapting the Continuous Knapsack Problem. Tripathi et al. employ commercial software [11], MetSim© and FactSage©; this approach is well-suited for simulation, but is not as well suited for mathematical optimization as the approach of Navarra et al. [53].

A particularly complicated aspect of Peirce-Smith thermochemistry is the representation of the cold charges. Marcuson et al. used industrial data to estimate the heat content and sensible heat [138], but did not estimate the thermal response parameters, unlike Navarra et al. [55]. Considering the importance of cold charging [66], the academic literature seems to be lacking reliable estimation techniques.

Thermochemical modeling of nickel-copper PSC had been considerably slower than for

copper PSC, until the work of Kellogg et al. in 1986 [139], describing the interactions of Ni, Fe, S, O, SiO_2 and CaO, in the formation of matte and slag. This work was soon followed by the work of Bustos et al. that describes the Falconbridge Smelter [140], and Kyllo et al. that describes the Copper Cliff Smelter [39, 52]; these works provide a blow-by-blow analysis, representing changes in bath temperature and composition. The later efforts of Kyllo et al. combined kinetics and thermodynamic considerations [57, 141], suggesting that productivity can be enhanced through changes in gas injection practices.

Tan et al. are also noted for their blending of kinetics and thermodynamics [134, 135, 142]. Particular emphasis has been placed on magnetite formation, as well as the bath volume and temperature. This work benefitted from the application of the Semtech technology, and has lead to improvements in the Mount Isa operations. The current model is capable of predicting the slag viscosity and liquidus temperature [142].

Partly as an effort to market the ALSI and other oxygen injection practices, a software system was developed by Ng et al. [10]. This software computed heat and mass balances, as well as operation costs, but only for copper PSC systems. A later adaptation by Navarra et al. provided more flexibility [66], and was eventually generalized for copper-nickel systems [53]. However, this approach was left wanting, as it did not include a sufficient description of operational aspects. Nonetheless, the generalization of the underlying model of Navarra et al. has been mimicked within in the current thesis (See Section 3.4 and Subsection 4.3.3).

Certain researchers have focused on the Copper-Blow [143, 144]. It was observed that surface tension effects cause the desulfurization to occur in two distinct stages: (1) the melt is partially desulfurized, as oxygen is dissolved in the matte; (2) after the bath is saturated with oxygen, SO_2 and copper are generated electrochemically [143]. It has also been found that the desulfurization rate is sensitive to the flow rate, but not so much to the temperature [144]. The resulting kinetics models compare well to experimental measurements [143, 144].

The successful use of the Peirce-Smith Converter depends on effective mixing [7]. The study of mixing has benefitted from advances in computational fluid mechanics [145, 146, 147]. These approaches apply the k-ϵ model, as a two-parameter representation of turbulence. Cervantes et al. presented an especially efficient approach, in which the converter is represented as a single transversal slice [145]; it is presumed that the equal-spacing of the tuyeres leads to periodic distribution of the velocity field.

The most advanced CFD models have not yet been integrated with the most advanced thermodynamics models. Perhaps future work will tap into this unresolved potential.

A.4 Failure to Adapt Conventional Scheduling Algorithms to PS Converting

Chemical thermodynamics and chemical kinetics remain are still regarded as the two main pillars of PSC modeling. The current thesis brings forth a third pillar that is as of yet under-represented: operational dynamics. This new pillar presents an untapped potential, especially when comparing current scheduling practices for PSC to those of the transportation and manufacturing industries [148, 149].

In Peirce-Smith Converting, the thermochemical complexity is germane to process control [66], and is perhaps the main factor that has obstructed schedule optimization. This is not to say that transportation and manufacturing are void of thermochemical complexity. For instance, an airliner considers fuel grades, combustion efficiencies, etc., but these thermochemical aspects can be decoupled from the problem of airline scheduling [148]. Early models for transportation and manufacturing were rather general, but were realistic enough to be integrated into industry [150]. Following these general models, more specialized models could very well include domain-specific elements, including thermochemical considerations. This path, from general to specialized, has not been replicated in PSC, because even the most basic descriptions must address some of the thermochemical complexities in order to have any industrial merit [47, 66].

Thermochemical complexity is not the only factor that has obstructed the development of algorithms in copper smelting. A comparison can be made to steelmaking, for example, in which optimal scheduling has been implemented [151]. The BOP and the PSC process are both batchwise processes that are fed by continuous streams, hence a clash between batch and continuous dynamics. However, the BOP process is relatively simple to manage, because the chemical composition of the feeds is essentially constant [152]. In contrast, PSC is noted for its particular ability to accept a variety of feeds [66], with tremendous variations in chemical composition. This is a functional strength of PSC, but it precludes the use of conventional scheduling algorithms. Thus the chemical variation of flow streams is an important secondary factor that distinguishes copper smelting from other chemical and materials industries, including steel.

At the current time, copper and nickel smelters commonly apply manual scheduling techniques for their short-term (daily) operations [47]. All of the start and finishing times are entered into the main computer system, which then transmits the resulting schedule to the control rooms. There are two main problems with this approach. Firstly, the schedules may not be optimal with respect to any particular objective. Consequently, there is no rigorous way of posing and analyzing alternative objectives. Secondly, the scheduling logic is a crit-

ical component of process simulation [12], which is used to quantify potential changes to a smelter. To evaluate an upgrade of the offgas treatment system, for example, it is helpful to simulate several months of operation, taking daily random variations into account. To avoid the task of manually scheduling hundreds of simulated days, the scheduling algorithms may be grafted into the simulations. If the scheduling practices are not automated, then it is difficult to justify plant upgrades, as it is not practical to simulate alternative plant designs over large ranges of time.

Nonetheless, very little work has been published regarding the optimal scheduling of Peirce-Smith operations [47]. This is especially surprising, considering that the profitability of a smelter is directly affected by its scheduling practice. Furthermore, the implementation of scheduling algorithms is not capital intensive.

A.5 Pyrometallurgical Alternatives to PS Converting

In addition to the Hoboken Converter (Subsection 1.3.3 and Appendix A.2), and the Noranda Converter (Appendix A.2), there are several pyrometallurgical alternatives to Peirce-Smith Converting for the treating copper and nickel-copper mattes.

Some of these alternatives are inspired by the steel industry. For instance, the Copper Cliff Smelter adapted a Kaldo unit in 1973 [153], which remained in use until 2002 [8]. "Kaldo" is the common name used in steelmaking, whereas the nonferrous industry uses the more descriptive terminology Top-Blown Rotary Converter (TBRC). These units have the upright shape of a ladle, except that the mouth is tapered around where the lance is lowered. They are used to melt and convert high-grade concentrates and metallic scrap. The TBRC unit is no longer in use in Copper Cliff, but is present at the Clydach Nickel Refinery [8], as well as the Stillwater Nickel Smelter [2].

Some researchers have investigated the possibility of a bottom-blown converting vessel, which would be similar to certain steelmaking vessels, but adapted to the necessities of matte conversion [154, 155]. Such converters have not yet been employed in the nonferrous sector, but have been the subject of CFD computations. Gonzalez et al. have employed the k-ϵ model to represent turbulence, in conjunction with the Volume-of-Fluid (VOF) formulation to track the interfaces. Apparently, the mixing characteristics of a bottom-blown matte converter would be favourable.

The Mitsubishi process was once seen as a major competitor to the Peirce-Smith Converter. In fact, the Mitsubishi process combines roasting, smelting and converting into a

single continuous process [8], which employs top-blown submerged oxygen injection. This integrated approach has a certain aesthetic, as there is no need to transport ladles from an independent smelting unit. Instead, the Mitsubishi system is composed of three furnaces, which are connected by a system of launders that are insulated and heated. This type of matte transportation leads to less fugitive emissions than the classic Peirce-Smith approach. Experience with the Mitsubishi process has shown that magnetite formation could be better managed by including a CaO component in the slag [156]. There are now only three remaining examples of the Mitsubishi process, following the recent closing of the Kidd Creek installation [1]. Furthermore, the sale of the Mitsubishi process has been suspended, which is partly related to the advent of other continuous converting technologies.

The Ausmelt Copper Converting and Isaconvert processes are taking the place of the Mitsubishi process [157, 158]. Both of these processes evolved from top-blown continuous smelting processes [8]. Both technologies feature a top-blown lance that is submerged into the melt, as blister copper is drained from the bottom of the vessel. As with the Mitsubishi Process, these technologies are noted for their use of CaO flux, and the effective control of fugitive emissions. Unlike the Mitsubishi Process, these vessels may be run in either a batch or a continuous more. There appears to be only a superficial difference between the Ausmelt and Isaconvert technologies, with regard to some aspects of the vessel geometry. However, the Ausmelt technology is more mature than the Isaconvert; the former has been installed in at least two commercial installations [157], whereas the Isaconvert is only in the early stages of commercial validation [158].

The Ausmelt and Isaconvert processes are likely candidates to supplant the Peirce-Smith Converter in the copper industry. However, the most compelling technology may be the Flash Converting Furnace (FCF), developed by Kennecott and Outotec. This technology builds on experience in flash smelting [26], which involves the rapid reaction of falling concentrate particles as they land into the melt. Flash converting employs the same principle as flash smelting, as solidified matte particles are dropped into the FCF. Thus the solidified (granulated) matte may be stockpiled as an intermediate product, composed mainly of chalcocite Cu_2S. The flash technology performs much better than Peirce-Smith converting, particularly in terms of refractory wear. While the kinetics of PSP relies on vigorous mixing in the melt, the FCF relies on the emersion of the falling particles within the countervailing blast. Due to the quiescent FCF bath, the refractory lining has a campaign life exceeding five years [26], which is twenty times longer than the campaign life of PSP [42]. Consequently, the FCF can operate at much higher oxygen enrichment, which is favourable for acid production. The FCF technology has been successfully applied at the Kennecott Copper Smelter

since 1995, and the Yanggu Xiangguang Copper Smelting since 2007 [26]. Incidentally, the Yanggu Xiangguand Copper Smelting is thought to be the most advanced copper smelter in the world.

Flash converting raises interesting questions for the copper supply chain. Chalcopyrite concentrate may not be the most viable product from copper sulfide mines. Clearly, the transportation of chalcopyrite $CuFeS_2$ from the mine to the smelter, is not as favourable as chalcocite Cu_2S transportation. Firstly, the iron component may be better disposed of at the mine sites, as fayalite and/or olivine can be incorporated into backfill material; current practices have led to enormous piles of slag outside of copper smelters, to the extent that entire towns have been buried in slag [160]. Aside from the slag disposal issues, part of the sulfur component of chalcopyrite may also be better utilized at mine sites, for the local processing of copper oxides. This aspect may be especially important for future mining operations, which are likely to have complex deposits, including various mixtures of copper oxides and sulfides. This would imply a blending of hydrometallurgical operations (for cathode production from oxides) with flotation and pyrometallurgical operations (for chalcocite and acid production from sulfides), all at a relatively small scale, and close to the mines.

Flash converting has also been implemented within the nickel industry, in the form of the Direct-Outotec-Nickel (DON). The DON technology was first installed at the Harjavalta Oy Smelter in 1995, and then at the Fortaleza Smelter in 1998 [26, 161]. This technology converts nickel-bearing concentrates directly into Bessemer matte, in a single process step, thus obviating the need for Peirce-Smith Converting. However, the DON technology does not replace the downstream operations that would release the valuable materials (See Subsection 1.2.5).

Given the advanced technologies that are being developed, it seems unlikely that the Peirce-Smith Converter will dominate indefinitely. For historical accounting, it will be interesting to document whether the reign of PSC will be longer than that of the Welsh process.

APPENDIX B

OVERVIEW OF MIXED INTEGER LINEAR PROGRAMMING

B.1 Linear Programming and the Simplex Method

Linear programming is an elementary form of constrained optimization. It has wide application in industrial and academic contexts, and is fundamental to management science.

Linear programming is a subclass of mathematical programming [162]. In this sense, "programming" is a somewhat antiquated use of the word that predates the advent of computers. This older notion of programming involves the creation of lists, tables, arrays, etc. to facilitate problem formulation and solution. Thus mathematical programming is the listing of variables, constraints, equations and other mathematical constructs, to which solution algorithms are applied. This does not necessarily imply the use of a computer, in principle. Nonetheless, computers have become ubiquitous in mathematical programming.

A linear program can be expressed as

$$\begin{aligned} \max_{x_j} f = \quad & c_\circ + \sum_{j=1}^{n} c_j x_j \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij}^{\text{lt}} x_j \le b_i \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\ & \sum_{j=1}^{n} a_{ij}^{\text{gt}} x_j \ge b_i^{\text{gt}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \\ & \sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j = b_i^{\text{eq}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{eq}}\} \end{aligned} \tag{6.24}$$

in which a_{ij}^{lt}, a_{ij}^{gt} and a_{ij}^{eq} are the constraint parameters, b_i^{lt}, b_i^{gt} and b_i^{eq} are the righthand parameters, and c_j are the objective weighting parameters. Equation B.1 may be called the general form of a linear problem [163, 164].

Thus the problem is to obtain values for $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ which maximize the linear objective function f, while satisfying the linear constraints. A solution $(x_1, x_2, \ldots, x_n)$ is said to be feasible if it satisfies the set of constraints. A nonlinear program may have a nonlinear objective, or nonlinear constraints, hence a generalization of linear programming.

Unless otherwise stated, a mathematical program is usually assumed to have only continuous numerical variables. Appendices B.4-5 consider the possibility of discrete variables.

Equation B.1 poses a maximization that is representative of numerous contexts. The objective is often to maximize production, or profit, by properly allocating limited resources or time. Nonetheless, the formulation could be adjusted to represent a minimization problem. Indeed, the minimization $\min_{x_j} f'$ is generally equivalent to the maximization $\max_{x_j}(-f')$, so that f may be taken simply as $f = -f'$. For example, the minimization of economic losses could be interpreted as the maximization of economic profit, given that a loss is effectively a negative profit.

The constant $c_\circ$ can be dropped from the objective, since it does not impact the optimality of any proposed solutions. This is to say that $(x_1, x_2, \ldots, x_n)$ is optimal for $(c_\circ + \sum_{j=1}^{n} c_j x_j)$ if, and only if, it is optimal for $\sum_{j=1}^{n} c_j x_j$. In the economic context, $c_\circ$ may be interpreted as some profit (or expense) which has already be assured, and need not be reexamined in the current decision-making process.

Linear programming incorporates the more basic concepts of systems of equations. If the objective f is constant, and if all of the constraints are equalities, $m^{\text{gt}} = m^{\text{lt}} = 0$, then Equation B.1 reduces to

$$\sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j = b_i^{\text{eq}} \tag{6.25}$$

for all $i \in \{1, 2, \ldots, m^{\text{eq}}\}$, which is a system of linear equalities. This is often written in matrix form,

$$A\mathbf{x} = \mathbf{b}$$

where A is a matrix containing a_{ij}^{eq} in the i^{th} row and j^{th} column; $\mathbf{x}$ and $\mathbf{b}$ are column vectors containing $(x_1, x_2, \ldots, x_n)$ and $(b_1^{\text{eq}}, b_2^{\text{eq}}, \ldots, b_{m^{\text{eq}}}^{\text{eq}})$, respectively. Equation B.2 is particularly relevant to extractive metallurgy because of the applications in mass and heat balancing (Section 3.2). However, the larger notions of linear program (Equation B.1) are not pervasive in extractive metallurgy, as it is not usually part of the undergraduate pedagogy.

It is always possible to reexpress the constraints exclusively as "$\leq$" inequalities. Firstly, the "$\geq$" inequalities can be converted simply by negating the parameters, hence

$$\left(\sum_{j=1}^{n} a_{ij}^{\text{gt}} x_j \geq b_i^{\text{gt}}\right) \Leftrightarrow \left(\sum_{j=1}^{n} a_{i'j}^{\text{gt}} x_j \leq b_{i'}^{\text{gt}}\right)$$

where $a_{i'j}^{\text{lt}} = -a_{ij}^{\text{gt}}$ and $b_{i'}^{\text{lt}} = -b_i^{\text{gt}}$. Additionally, each equality can be replaced by two inequal-

ities,

$$\left(\sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j = b_i^{\text{eq}}\right) \Leftrightarrow \begin{pmatrix} \sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j \leq b_i^{\text{eq}} \\ \\ \sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j \geq b_i^{\text{eq}} \end{pmatrix}$$

Thus an equality is essentially a combination of a "$\leq$" inequality and a '$\geq$" inequality, the latter of which can ultimately be converted into a second "$\leq$" inequality through negation. This procedure leads to $m = (m^{\text{lt}} + m^{\text{gt}} + 2m^{\text{eq}})$ constraints.

There is a more effective means of handling the equalities. Of the m^{eq} equalities in Equation B.1, a certain quantity of these will be linearly independent [165], $m^{\text{ind}} \leq m^{\text{eq}}$; thus m^{ind} variables can be eliminated by substitution, reducing the size of the problem. Thus Equation B.1 is reduced to

$$\begin{aligned} \max_{x_j} f = \quad & \sum_{j=1}^{n} c_j x_j \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m\} \end{aligned} \tag{6.26}$$

in which a maximal number of variables has been eliminated, and $m = m^{\text{lt}} + m^{\text{gt}}$. Equation B.3 can be referred to as "the" reduced form, although there is some ambiguity regarding which variables are to be eliminated. Nonetheless, the eliminated variables can be computed *a posteriori*, after the reduced problem has been solved.

For instance, suppose that a list of variables (x_1, x_2, x_3) is defined as the basis for a decision-making policy; a researcher may pose the following linear program,

$$\begin{array}{rrcrcrcr} \max f & = & 2x_1 & + & 4x_2 & - & 3x_3 & + & 7 \\ \\ \text{such that} & & x_1 & - & x_2 & & & \leq & 3 \\ & & x_1 & & & & & \leq & 5 \\ & & x_1 & + & x_2 & + & 2x_3 & \leq & 6 \\ & & 3x_1 & & & - & x_3 & \geq & 7 \\ & & & & x_2 & - & x_3 & = & 6 \end{array}$$

The equality gives $x_3 = x_2 - 6$, so that a reduced form can be obtained in terms of (x_1, x_2),

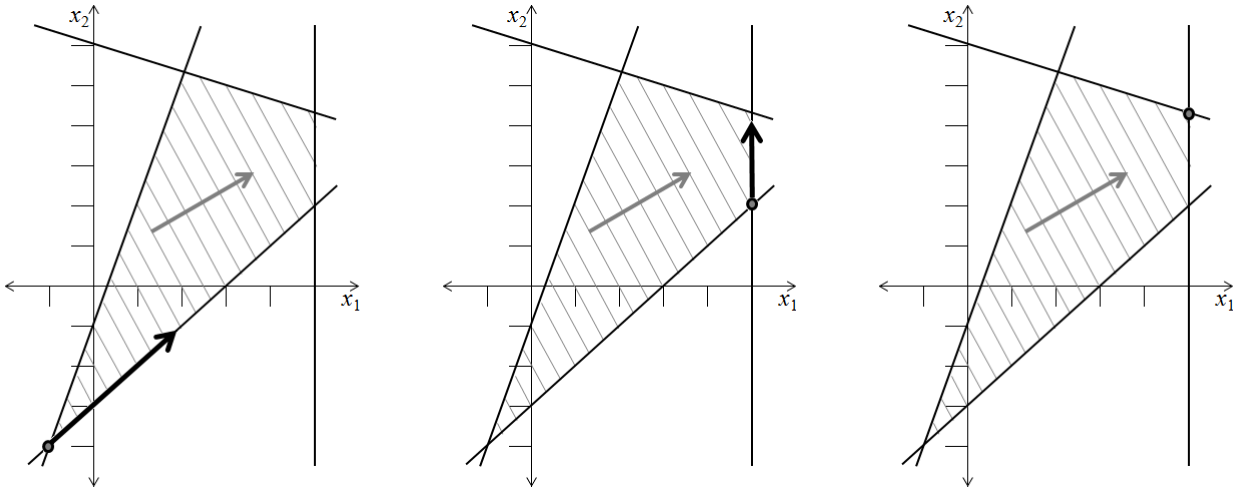
$$\begin{array}{rcrcrcl} \max f & = & 2x_1 & + & x_2 & & \\ & & & & & & \\ \text{such that} & & x_1 & - & x_2 & \leq & 3 \\ & & x_1 & & & \leq & 5 \\ & & x_1 & + & 3x_2 & \leq & 18 \\ & & -3x_1 & + & x_2 & \leq & -1 \end{array}$$

After the optimal solution (x_1^*, x_2^*) has been determined, the corresponding value for x_3 is computed, $x_3^* = x_2^* - 6$.

Historically, it was important to categorize the various forms of linear programs, from which algorithms would be developed. The reduced form (Equation B.3) is insightful, particularly for two-dimensional problems (Figure B.1). If a 2D linear program possesses optimal solutions, then at least one of these optimal solutions can be observed at a vertex; this can be proven formally, using the properties of convex polygons. It is thus sufficient to search only the feasible vertices.

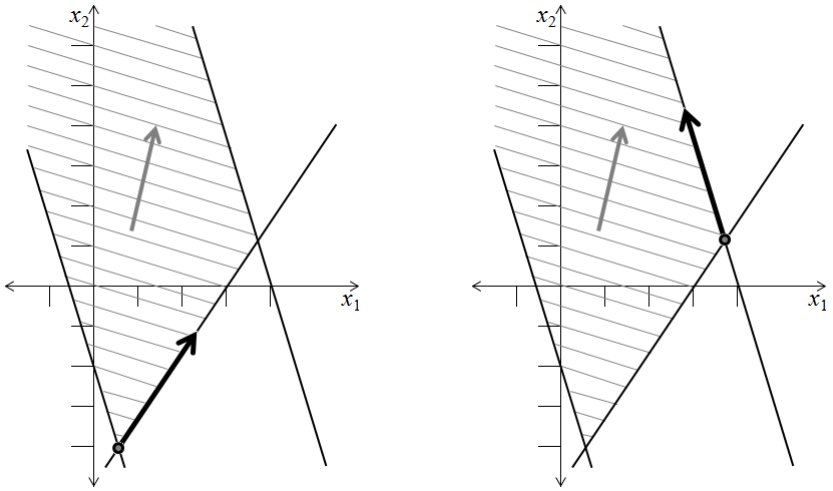
Figure B.1 demonstrates the iterative procedure by which vertices are searched, always selecting directions of steepest ascent; the objective function is depicted through the use of level curves. The feasibility boundary forms rays that connect one vertex to the next, and optimality is detected at vertices which do not offer any directions of further improvement. Figure B.1 demonstrates the migration from the initial vertex (-1,-4) to the optimal vertex $(5, \frac{13}{3})$, requiring two iterations. Figure B.2 depicts a case where the objective is unbounded. This is detected when the direction of steepest ascent proceeds indefinitely along a constraint line, uninterrupted by any of the other constraints.

In 1947, George Dantzig generalized the procedure of Figures B.1-2 to n variables [166], which is one of the greatest advancement in 20^{th} applied mathematics [167]. The formal justification is based on the properties of convex polytopes, which are a generalization of convex polygons. In two dimensions, each iteration analyzes the intersection of two incident lines, which form the tip of a triangle (Figure B.3a); in three dimensions, each iteration analyzes the intersection of three incident planes, which form the tip of a tetrahedron (Figure B.3b); in n dimensions, each iteration analyzes the intersection of n incident hyperplanes, which form the tip of simplex (i.e. the n-dimensional extension of triangles and tetrahedra). Dantzig's procedure is therefore known as the Simplex Method.



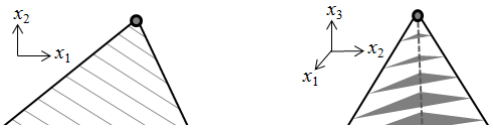
(a) Suboptimal (b) Suboptimal (c) Optimal

Figure 6.4: Application of the Simplex Method for two decision variables (x_1, x_2), leading to an optimal solution $(x_1^*, x_2^*) = (5, \frac{13}{3})$



(a) Suboptimal (b) Suboptimal

Figure 6.5: Application of the Simplex Method for two decision variables (x_1, x_2), leading to an unbounded objective value



(a) 2D (b) 3D

Figure 6.6: Vertices are surrounded by simplex neighbourhoods

The Simplex Method has been revised and refined to minimize the number of arithmetic operations [168, 169]. Matrix algebra has been adapted to efficiently compute the intersections of lines, planes, hyperplanes, etc., to locate and analyze the vertices. Nonetheless, there are several introductory texts that show how to solve small linear programs manually, using of Simplex Tableaus [170, 171]; students can then compare their manual results to the computed results, before attempting larger problems.

An apparent complication with the Simplex Method is the requirement for an initial vertex that would allow the first iteration [163, 164, 170, 171]. In some cases, an initial vertex can be deduced from the particular structure of the problem. Otherwise, an auxiliary problem can be posed,

$$\begin{aligned} \max_{x_j,\alpha_i} f = \quad & \sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} \alpha_i \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij} x_j - \alpha_i \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m\} \end{aligned} \tag{6.27}$$

which is called the Big-M Formulation, in which α_i are artificial variables, and $M > 0$ is the cost associated to these variables, described below. A vertex is readily available for Equation B.4, and is given by $(x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$ and $(\alpha_1, \alpha_2, \ldots, \alpha_m) = (b_1, b_2, \ldots, b_m)$.

The artificial cost M must be large enough so that the objective function is dominated by the artificial component $M \sum_{i=1}^{m} \alpha_i$. In a sense, it is infinitely costly for a solution to be infeasible. Thus, the first m Simplex iterations of Equation B.4 should evolve toward a vertex solution such that $(\alpha_1, \alpha_2, \ldots, \alpha_m) = (0, 0, \ldots, 0)$, hence eliminating the artificial variables; the corresponding non-artificial values $(x_1, x_2, \ldots, x_n)$ can then serve as an initial vertex for Equation B.3. If the Simplex iterations fail to eliminate the artificial variables, then Equation B.3 is infeasible, meaning that the constraints are contradictory. Thus the artificial formulation (Equation B.4) tests for the feasibility of the original problem (Equation

B.3), as well as providing an initial vertex.

The elimination of the artificial variables is frequently referred to as Phase I of the Simplex Method, and the subsequent solution of Equation B.3 is then Phase II. For Phase I, the objective of Equation B.4 may be rescaled by dividing through by M; taking the appropriate limit $M \to \infty$,

$$\max_{x_j,\alpha_i} f_{\text{PhaseI}} = \lim_{M\to\infty} \left(\frac{\sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} \alpha_i}{M} \right)$$

hence removing any doubt that M would be large enough,

$$\max_{x_j,\alpha_i} f_{\text{PhaseI}} = -\sum_{i=1}^{m} \alpha_i \tag{6.28}$$

Upon completion of Phase I, the resulting $(x_1, x_2, \ldots, x_n)$ solution serves as an initial vertex for Equation B.3, as before.

In industrial contexts, the Simplex Method is commonly implemented within automated decision-making software. This has applications in engineering design, systems management, and operational scheduling.

B.2 Alternatives to the Simplex Method

The Simplex Method is historically the most important procedure for solving linear programs. However, there are cases where the Simplex Method is known to perform inefficiently.

As part of a worst-case analysis [172], particular problems have been constructed to demonstrate that the computational costs can grow exponentially as a function of the problem dimensions (m, n). This prompted the development of the Ellipsoid Method [173], which was presented in 1979, and is proven to have a polynomial expansion in the worst-case.

In spite of a more favourable worst-case analysis, the Ellipsoid Method did not perform as well as the Simplex Method in "real-world" problems, hence it was never widely adopted [174]. On the other hand, it inspired the Projective Method that was presented in 1984, which has polynomial expansion comparable to the Ellipsoid Method, and performs considerably better than the Ellipsoid Method [175, 176].

The Projective Method is considered to be the first of a family of algorithms, called the

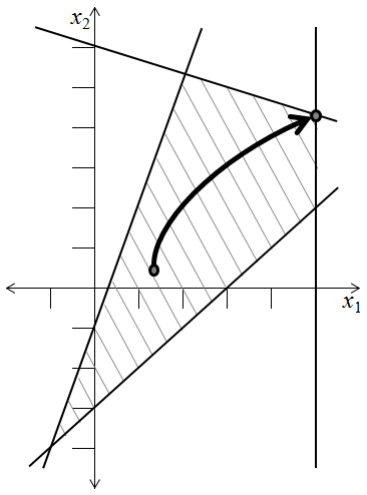


Figure 6.7: Path-Finding Methods approach optimal solutions from the interior of the feasibility region

Interior Point Methods. In general, these methods apply numerical techniques to advance from the the interior of the feasibility region toward the optimal vertex [174]. This is unlike the Simplex Method, which is restricted to the outer vertices.

Starting in the 1990's, the most successful Interior Point Methods have been the Path-Following Methods [174]. As depicted in Figure B.4, these methods migrate along a path, which ultimately leads to the optimal vertex. This is usually accomplished using a barrier function, which penalizes the current solution for its distance to the feasibility boundary; thus, the interior points are penalized for being *too interior*. As each iteration progresses, the barrier function becomes increasingly severe, hence promoting solutions that are increasingly closer to the boundary. The optimal vertex is eventually identified to within computer precision.

An appropriate barrier function $B^i(x_1, x_2, \ldots, x_n)$ should, to some degree of approximation, satisfy the following property,

$$\lim_{i\to\infty} B^i(x_1, x_2, \ldots, x_n) = \begin{cases} B & \text{if } (x_1, x_2, \ldots, x_n) \text{ is on the boundary of the feasibility region} \\ \\ \infty & \text{otherwise} \end{cases} \tag{6.29}$$

where $i \in \{1, 2, \ldots\}$ denotes the iteration number, and B is a finite constant. The barrier function can always be adjusted so that $B = 0$, but this is not strictly necessary.

Following this construction, the Interior Point Methods solve the following mathematical

program at each iteration i'.

$$\begin{aligned} \max_{x_j} f = \quad & \sum_{j=1}^{n} c_j x_j - B^{i'}(x_1, x_2, \ldots, x_n) \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m\} \end{aligned} \tag{6.30}$$

This is a nonlinear program, as the barrier function is generally nonlinear. Path-Finding Methods are distinguished from each other in the type of barrier function, and the numerical procedure that is used at each iteration to solve Equation B.7.

The barrier function of Equations B.7 acts as the dual of the artificial cost function of Equation B.4. More precisely, the barrier function penalizes a solution for being strictly within the feasibility region, thus approaching the boundary from the interior; conversely, the artificial costs penalize solutions for their (strict) infeasibility, thus approaching the boundary from the exterior. This notion of duality is revisited in the following section.

The development and adaptation of Interior Point Methods is an active area of research. There is now a consensus that modern Path-Following Methods have a similar performance as the Simplex Method for routine applications, in addition to the favourable worst-case analysis [177]. For specific problems, however, the Simplex Method may perform better, and sometimes much better [178].

B.3 Expansion of a Solved Linear Program

The Simplex Method has a historical advantage over Path-Following Methods. In particular, the Simplex Method is amenable to the inclusion of new linear variables.

To include a new variable x_{n+1}, Equation B.3 is modified only slightly.

$$\begin{aligned} \max_{x_j} f = \quad & \sum_{j=1}^{n} c_j x_j + c_{n+1} x_{n+1} \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij} x_j + a_{i,n+1} x_{n+1} \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m\} \\ & x_{n+1} \leq x_{n+1}^{\text{Initial}} \end{aligned} \tag{6.31}$$

The new variable x_{n+1} should be formulated so that $x_{n+1} = x_{n+1}^{\text{Initial}}$ can be treated tentatively as

a constant parameter, and the final constraint can be ignored, as it is automatically satisfied. Moreover, $c_{n+1}x_{n+1}$ may be temporarily dropped from the objective, and $a_{i,n+1}x_{n+1}$ can be temporarily fused into the righthand constants that become effectively $b_i - a_{i,n+1}x_{n+1}$. An initial optimization will obtain values for $(x_1, x_2, \ldots, x_n)$, leaving $x_{n+1} = x_{n+1}^{\text{Initial}}$ as a constant. Subsequently, $(x_1, x_2, \ldots, x_n, x_{n+1}^{\text{Initial}})$ serves as an initial vertex for Equation B.8, to proceed directly into Phase II of the Simplex Method.

The approach of Equation B.8 is to perform post-optimization, looking for beneficial decreases in x_{n+1}. Beneficial increases can be examined if the final inequality of Equation B.8 is replaced with

$$-x_{n+1} \leq -x_{n+1}^{\text{Initial}} \tag{6.32}$$

which still maintains the reduced form (Equation B.3).

Equations B.8-9 allow the *ad hoc* expansion of a linear program from (m, n) to $(m, n+1)$. Historically, this has relied on the Simplex Method's ability to pass seamlessly from the vertices of a space in $\mathbb{R}^n$, onto the neighbouring vertices of a superspace in $\mathbb{R}^{n+1}$. Naturally, the Path-Following Methods have been adapted to provide a similar feature [179].

The inclusion of a new variable is an expansion of the feasibility region into new dimensions. This new degree of freedom would tend to improve the objective value. In an extreme case, the expanded $(m, n+1)$ problem may be unbounded (infinite production, infinite profits, etc.), which may lack some realism, unless additional constraints can also be added. Following the example of Equations B.8-9, it is beneficial if the solution to the original (m, n) problem can somehow act as an initial vertex for the expanded $(m+1, n)$ problem, hence passing directly into Phase II of the Simplex Method.

The following formulation is related to Equation B.1.

$$\begin{aligned} \min_{y_i} g = & \sum_{i=1}^{m^{\text{lt}}+m^{\text{gt}}+m^{\text{eq}}} b_i y_i \\ \text{such that } & \sum_{i=1}^{m^{\text{lt}}} a_{ij} y_i + \sum_{i=1}^{m^{\text{gt}}} a_{ij}^{\text{gt}} y_i + \sum_{i=1}^{m^{\text{eq}}} a_{ij}^{\text{eq}} y_i \leq c_j \qquad \text{for all } j \in \{1, 2, \ldots, n\} \\ & y_i \geq 0 \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\ & y_i \leq 0 \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \end{aligned} \tag{6.33}$$

Equation B.1 is the primal formulation, whereas Equation B.10 is the corresponding dual formulation [180]. It can be verified that the dual of the dual is the primal, although the

irrelevant constant term $c_\circ$ must be dropped. This procedure can be applied to a system of equalities (Equation B.2), so that

$$\sum_{i=1}^{m^{\text{eq}}} a_{ij}^{\text{eq}} y_i = c_j \tag{6.34}$$

for all $j \in \{1, 2, \ldots, n\}$, or equivalently

$$A^T \mathbf{y} = \mathbf{c}$$

in which A^T is the transpose of A, $\mathbf{y}$ is the vector of dual variables, and $\mathbf{c}$ is the objective weighting vector. The dual is hence a transposition between variables and constraints. Adding a new primal constraint is equivalent to adding a new dual variable.

The Duality Theorem states that if the primal problem (Equation B.1) has an optimal solution $(x_1^*, x_2^*, \ldots, x_n^*)$, then the dual problem (Equation B.10) has an optimal solution $(y_1^*, y_2^*, \ldots, y_m^*)$ such that

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^* \tag{6.35}$$

Moreover, primal solutions $(x_1, x_2, \ldots, x_n)$ are directly associated to dual solutions $(y_1, y_2, \ldots, y_m)$, such that

$$\left(\sum_{j=1}^{n} a_{ij} x_j < b_i \right) \Rightarrow (y_i = 0) \tag{6.36}$$

for all $i \in \{1, 2, \ldots, m^{\text{lt}}\}$,

$$\left(\sum_{j=1}^{n} a_{ij}^{\text{gt}} x_j > b_i \right) \Rightarrow (y_i = 0) \tag{6.37}$$

for all $i \in \{1, 2, \ldots, m^{\text{gt}}\}$. Equations B.13-14 are the Complementary Slackness Conditions [180].

The dual variables y_i are sometimes called Lagrangian multipliers, and they quantify the "force" that a constraint is exerting against the objective function that is "driving" the optimization; indeed, duality theory is tied to Lagrangian mechanics [181], in the notion of generalized forces (e.g. forces, torques, pressures). Equations B.13-14 describe solutions for which the constraints are slack, hence do not exhibit a counteracting force, $y_i = 0$. Duality is especially noted in game theory, but has applications in physics, economics, etc., insomuch as it is an inherent feature of constrained optimization [182]. Duality theory has lead to major advancements in Interior Point Methods, for both linear and nonlinear programming [174, 183].

There are some implementations of the Simplex Method that can alternate between the primal and dual formulations. Essentially, a primal Simplex iteration penalizes a feasible vertex for being suboptimal, whereas a dual Simplex iteration penalizes a vertex for being infeasible. Elementary texts in mathematical programming demonstrate how to effectuate dual Simplex iterations using Simplex Tableaus; primal iterations correspond to row operations, whereas dual iterations correspond to column operations [171, 184].

In relation to the reduced form (Equation B.3),

$$\begin{aligned} \max_{y_i} g = \quad & -\sum_{i=1}^{n} b_i y_i \\ \text{such that} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j \qquad \text{for all } j \in \{1, 2, \ldots, n\} \\ & -\sum_{i=1}^{m} a_{ij} y_i \leq -c_j \qquad \text{for all } j \in \{1, 2, \ldots, n\} \end{aligned} \tag{6.38}$$

Equation B.13 is the dual of the primal general form (Equation B.3), which itself has been placed in reduced form. The expansion Equation B.3, from (m, n) to $(m + 1, n)$, coincides with the introduction of a new dual variable y_{m+1} in Equation B.13, following the procedure developed in Equations B.8-9.

A new dual variable y_{m+1} augments the dual feasibility region, and can lead to an unbounded dual objective value. This extreme case coincides to the inclusion of a new constraint in the primal problem, $\sum_{j=1}^{n} a_{i,m+1} x_j \leq b_{\mathrm{m+1}}$, which conflicts with the existing constraints, rendering the primal problem infeasible. More generally, new constraints tends to diminish the primal objective, as they enhance the dual objective.

When new variables are included in Equation B.3, a sequence of primal simplex iterations can be used to reestablish optimality for the newly expanded problem. When constraints are appended to Equation B.3, dual simplex iterations can be used to reestablish feasibility for the newly expanded problem. In either case, the procedure drives toward solutions that are both feasible and optimal for the expanded problem.

The procedures for including new variables and new constraints into Equation B.3 can be adapted to Equation B.1, considering that Equation B.1 can always be converted into general form. Commercial platforms automatically execute these types of transformations [71], so they do not usually pose any practical concern.

B.4 Incorporation of Integer Variables

The type of problems that can be treated by linear programming is vastly increased if discrete variables and constraints are permitted. For example, a smelter design team may need to decide how many converters to install; they may include 3 or 4, but obviously there is no option to install 3.86 converters.

Equation B.1 is thus extended,

$$\begin{aligned}
\max_{x_j} f = \quad & c_\circ + \sum_{j=1}^{n} c_j x_j \\
\\
\text{such that} \quad & \sum_{j=1}^{n} a_{ij}^{\text{lt}} x_j \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{gt}} x_j \geq b_i^{\text{gt}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j = b_i^{\text{eq}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{eq}}\} \\
& x_j \in \mathbb{Z} \qquad \text{for all } j \in \{1, 2, \ldots, n_{\text{Int}}\}
\end{aligned} \tag{6.39}$$

where $n_{\text{Int}} \leq n$ is the number of variables that are under integrality constraints. Without loss of generality, Equation B.16 imposes an ordering on the variables, such that $(x_1, x_2, \ldots, x_{n_{\text{Int}}})$ have direct integrality constraints, whereas $(x_{(n_{\text{Int}}+1)}, x_{(n_{\text{Int}}+2)}, \ldots, x_n)$ are not directly placed under the integrality constraints.

Binary variables are a particular case of integer variables, which only consider two possible values, zero and one. For example, $x_1 \in \{0, 1\}$ can be represented within Equation B.16, using

$$\begin{aligned}
x_1 &\geq 0 \\
x_1 &\leq 1 \\
x_1 &\in \mathbb{Z}
\end{aligned}$$

Binary variables are sometimes called 0-1 variables [185]. As described in Appendix B.5, they are particularly useful for incorporating categorical variables.

An integer linear program (ILP) is usually understood to have $n_{\text{Int}} = n$, meaning that all variables are directly under integrality constraints; to be more explicit, the phrase "pure

integer linear program" may be used. In contrast, a mixed integer linear program (MILP) is such that $n_{\text{Int}} < n$. An MILP is said to include a mixture of integer and continuous variables.

To solve Equation B.16, the first wide-reaching success was due to the Branch-and-Bound Method [186], proposed in 1960. In this approach, a first approximation $(x_1^1, x_2^1, \ldots, x_n^1)$ may be obtained by simply ignoring the integer requirements, and applying one of the methods described in Appendices B.1-2; if this solution happens to satisfy the integrality constraints, then $(x_1^*, x_2^*, \ldots, x_n^*) = (x_1^1, x_2^1, \ldots, x_n^1)$ and the algorithm ceases. Otherwise, one of the violated constraints $j_1 \in \{1, 2, \ldots, n_{\text{Int}}\}$ is selected as a branching node, from which two more linear programs will be considered. Both of these new linear programs consider the constraints of the original (parent) linear program. However one of the new programs is extended with

$$x_{j_1} \leq \lfloor x_{j_1}^1 \rfloor$$

and the other is extended with

$$x_{j_1} \geq \lceil x_{j_1}^1 \rceil$$

This branching process has the effect of shrinking the feasibility region to eliminate the current nonintegral solution from further consideration in the subsequent branches.

In general, the k^{th} node is evaluated as a if it were a continuous linear program, free of any integrality constrains, giving an optimal solution $(x_1^k, x_2^k, \ldots, x_n^k)$. If some of the integrality conditions are not satisfied, then one of the violated constraints $j_k \in \{1, 2, \ldots, n_{\text{Int}}\}$ is selected as a branching node. One of the subsequent evaluations is extended with

$$x_{(j_k)} \leq \lfloor x_{(j_k)}^k \rfloor \tag{6.40}$$

and the other is extended with

$$x_{(j_k)} \geq \lceil x_{(j_k)}^k \rceil \tag{6.41}$$

Following the evaluation of a node, it is often unclear whether Equation B.17 should be explored first, or whether it should be B.18. In either case, these subsequent nodes would typically lead to more branching before encountering a solution that would satisfy all of the integrality constraints of Equation B.16.

To select the branching variables, a simple and common heuristic is given,

$$j_k = \underset{j \in \{1,2,\ldots,n_{\text{Int}}\}}{\text{argmax}} \; (0.5 - |x_j - \lfloor x_j \rfloor - 0.5|) \tag{6.42}$$

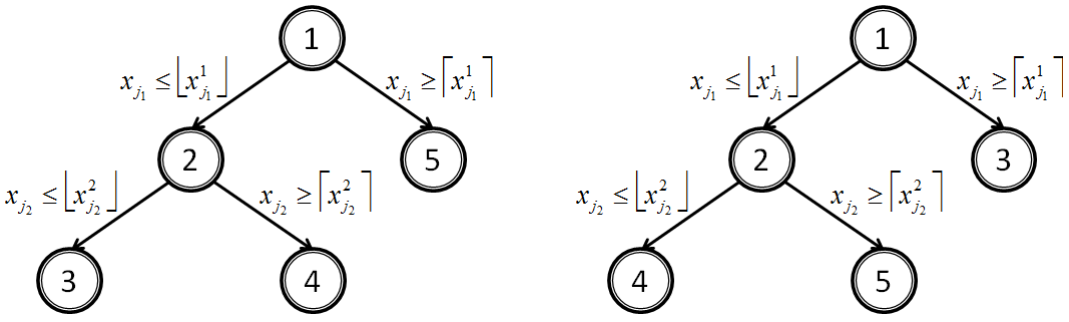
The idea is to select a variable for which it is most unclear whether it should be rounded up or rounded down. There are better performing branching heuristics than Equation B.19, both for general and specialized problem structures [187].

Each node represents a continuous approximation of Equation B.16, with an additional set of constraints that direct the search toward the integer optimum. As discussed in Appendix B.3, duality theory allows the constraints (Equations B.17-18) to be appended to the parent problem, without having to reinitiate the Simplex Method. For instance, $(x_1^1, x_2^1, \ldots, x_n^1)$ is used as an initial vertex to obtain the second approximation $(x_1^2, x_2^2, \ldots, x_n^2)$.

Figure B.5 illustrates the branching sequence, assuming that Equation B.17 always precedes Equation B.18. The nodes may be processed using a depth-first search (DFS) or a breadth-first search (BFS); commercial implementations usually combine these two approaches. DFS is often employed first, to quickly find and record a feasible solution that satisfies all of the integrality conditions, although it may not be the optimal solution. As the algorithm progresses, better feasible solutions are encountered, and are recorded in place of the incumbent feasible solution.

In the context of a maximization (Equation B.16), the incumbent solution provides a lower bound on the objective. Thus all of the nodes which offer no hope of surpassing the incumbent solution are immediately eliminated from future computations. A similar logic can be established in the context of a minimization, whence the current solution offers an upper bound.

As the algorithm progresses deeper into the tree, there is an increasing number of constraints (Equations B.17-18) which are accumulated. In many cases, the linear program is



(a) Depth-First Search (b) Breadth-First Search

Figure 6.8: Possible branching sequences to solve an MILP

found to be infeasible as the new constraint is appended. In such a case, the current node does not provide an integer solution, nor does it lead to any further branching.

In summary, there are three ways in which a node would be dismissed from further branching,

- The node produces a solution which happens to satisfy all of the integrality constraints, hence there is no variable x_{j_k} to form the branching. (If the corresponding objective value is superior than the incumbent objective value, then current solution becomes the new incumbent solution).
- The current node yields an objective value that is surpassed by the incumbent objective value, hence offering no hope of improving the current objective value.
- The current node has been fully constrained to the extent that the continuous problem is infeasible.

Every time that a new constraint is appended, dual simplex iterations are applied to detect if the new problem is infeasible, otherwise to reestablish the feasibility.

Following the initial development of the Branch-and-Bound Method, several other refinements have been incorporated, leading to the Branch-and-Cut Methods [185]. These methods apply a more aggressive processing of the nodes, before branching. This involves the inclusion of additional constraints (cuts) which are computationally beneficial; they are designed to shrink the feasible domain of the continuous problem of the node, without eliminating any of the integer feasible solutions of B.16.

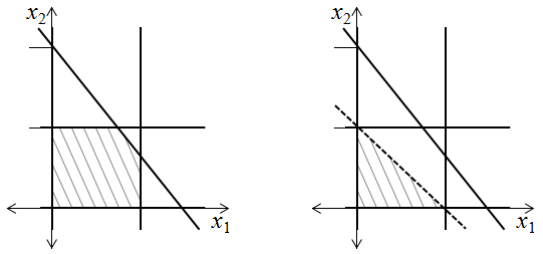
For example, the problem may include a linear constraint,

$$4x_1 + 3x_2 \leq 6$$

and the further restriction that x_1, x_2 are both binary (Figure B.6). Given that $x_1 = x_2 = 1$ would violate the constraint, it follows that no more than one of $(x_1 = 1)$ or $(x_2 = 1)$ can be true, hence the following cut can be added

$$x_1 + x_2 \leq 1$$

Figure B.6 shows that this additional cut eliminates two vertices which do not satisfy the integrality conditions, $(\frac{1}{2}, 1)$ and $(1, \frac{1}{2})$. The elimination of these two vertices may reduce the number of Simplex iterations, hence accelerating the computation. Figure B.6 is an example of the Gomory-Chvátal cutting procedure [188, 189], which was first presented in 1958.



(a) Before cut (b) After cut

Figure 6.9: Application of a cut, as part of the Branch-and-Cut Method

In addition to the Gomory-Chvátal cuts, several other procedures have been established for automatically generating appropriate cuts, and are applied successively to individual nodes, prior to proceeding with the branching operation. Once again, dual Simplex iterations are essential for the execution of these cuts.

A considerable effort has been devoted to developing branching protocols, particularly for scheduling problems [190, 191]. The Beam Search Methods perform a restricted search of the Branching-and-Bound tree, in which only the most promising nodes are prioritized. These promising nodes are explored via typical Branch-and-Cut procedures, and are thus part of the beam, i.e. they are "illuminated"; the rejected nodes remain unexplored, and are hence "dark". The criteria for exploring or rejecting a node may differ from one method to another. In general, these criteria are more severe than the simple comparison to the incumbent solution; thus a beam search would reject more nodes than a basic Branch-and-Cut approach, leaving the theoretical possibility that the resulting schedule would be suboptimal. In practice, the Beam Search Methods perform well, producing optimal or nearly optimal solutions, with high computational efficiency.

Branch-and-Cut Methods may differ in the type of cuts, as well as the branching protocols. These methods are now the dominant technique for solving MILP's, and continue to be an active area of research and development [185].

B.5 Incorporation of Categorical Variables

A particularly powerful feature of mixed integer linear programming is its capability to represent categorical (nonnumerical) variables. This is accomplished through the use of binary variables.

For example, there may be some categorical variable Var which may take certain values which are nonnumerical, Var $\in \{\text{A, B, C}\}$. Thus Var may be equal to A, B or C, but cannot be simultaneously equal to more than one of these values. This variable can be included into an MILP, firstly by defining the following binary variables,

$$\beta_{\text{Var,A}} = \begin{cases} 1 & \text{if Var} = \text{A} \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{\text{Var,B}} = \begin{cases} 1 & \text{if Var} = \text{B} \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{\text{Var,C}} = \begin{cases} 1 & \text{if Var} = \text{C} \\ 0 & \text{otherwise} \end{cases}$$

Secondly, these integer variables must be implemented along with the following restriction,

$$\beta_{\text{Var,A}} + \beta_{\text{Var,B}} + \beta_{\text{Var,C}} = 1$$

which prevents Var from taking on more than one value.

More generally, an MILP may include a set of categorical variables, Var^i for $i \in \{1, 2, \ldots, n_{\text{cvar}}\}$, where n_{cvar} is the number of categorical variables. Each of these variables is associated to a discrete range of values $\mathcal{V}$, which is implemented through the use of binary variables,

$$\beta^i_{\text{Var}k} = \begin{cases} 1 & \text{if Var}^i = k \\ 0 & \text{otherwise} \end{cases}$$

for all $i \in \{1, 2, \ldots, n_{\text{cvar}}\}$ and all $k \in \mathcal{V}$, in conjunction with the equality

$$\sum_{k \in \mathcal{V}} \beta^i_{\text{Var}k} = 1 \tag{6.43}$$

for all $i \in \{1, 2, \ldots, n_{\text{cvar}}\}$.

Certain formulations may consider a special value, Null $\in \mathcal{V}$, which allows Equation B.20 to be replaced by

$$\sum_{k \in \mathcal{V} \setminus \{\text{Null}\}} \beta^i_{\text{Var}k} \leq 1 \tag{6.44}$$

for all $i \in \{1, 2, \ldots, n_{\text{cvar}}\}$. This approach makes it unnecessary to implement the binary variables $\beta^i_{\text{Var,Null}}$. Indeed, $\text{Var}^i =$ Null if and only if the lefthand side of Equation B.20 is

zero.

Categorical variables can be used in combination with numerical variables. In particular, an MILP can support the following logical constraints,

$$\left(\text{Var}^i = k\right) \Rightarrow \left(\underline{x}_j^{ik} \leq x_j \leq \overline{x}_j^{ik}\right)$$

which relates Var^i to the numerical variable x_j. The lower and upper bounds, $\underline{x}_j^{ik}$ and $\overline{x}_j^{ik}$, must be respected when $\text{Var}^i = k$. These types of relationships are known as disjunctive bounds, and can be implemented using the following inequalities,

$$x_j \geq \sum_{k \in \mathcal{V}} \underline{x}_j^{ik} \beta_{\text{Var}k}^i \tag{6.45}$$

$$\leq \sum_{k \in \mathcal{V}} \overline{x}_j^{ik} \beta_{\text{Var}k}^i \tag{6.46}$$

for all $i \in \{1, 2, \ldots, n_{\text{cvar}}\}$, $j \in \{1, 2, \ldots, n\}$ and $k \in \mathcal{V}$.

A slightly more complicated structure is given by

$$\left(\text{Var}^i = k\right) \Rightarrow \left(\underline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \underline{a}_{jj'}^{ik} x_{j'} \leq x_j \leq \overline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \overline{a}_{jj'}^{ik} x_{j'} \right)$$

These are disjunctive linear constraints. In particular, if $\underline{a}_j^{ik} = \overline{a}_j^{ik} = a_j^{ik}$ and $\underline{a}_{jj'}^{ik} = \overline{a}_{jj'}^{ik} = a_{jj'}^{ik}$ for all j', then the condition reduces to

$$\left(\text{Var}^i = k\right) \Rightarrow \left(x_j = a_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} a_{jj'}^{ik} x_{j'} \right)$$

Disjunctive linear constraints are implemented into an MILP through the following inequal-

ities,

$$x_j \;\geq\; \underline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \underline{a}_{jj'}^{ik} x_{j'} - \left(\overline{x}_j - \underline{x}_j\right)\left(1 - \beta_{\mathrm{Var}k}^{i}\right) \tag{6.47}$$

$$\leq\; \overline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \overline{a}_{jj'}^{ik} x_{j'} + \left(\overline{x}_j - \underline{x}_j\right)\left(1 - \beta_{\mathrm{Var}k}^{i}\right) \tag{6.48}$$

for all $i \in \{1, 2, \ldots, n_{\mathrm{cvar}}\}$, $j \in \{1, 2, \ldots, n\}$ and $k \in \mathcal{V}$. Sometimes $k = \text{Null}$ is treated as a special case. The global bounds, $\underline{x}_j$ and $\overline{x}_j$, must be set such that $\underline{x}_j \leq x_j \leq \overline{x}_j$, regardless of the value of Var^i. It is generally helpful that $\underline{x}_j$ be as large as possible, and $\overline{x}_j$ be as small as possible, while respecting the condition that $\underline{x}_j \leq x_j \leq \overline{x}_j$; much like the cutting procedures described in Appendix B.4, this tends to diminish the relaxed (continuous) feasibility region, hence accelerating the computations. When x_j is a nonnegative variable, it is often appropriate to use $\underline{x}_j = 0$.

If Var^i is of a particular disjunctive category k, then $\beta_{\mathrm{Var}k}^{i} = 1$, and Equations B.24-25 become

$$x_j \;\geq\; \underline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \underline{a}_{jj'}^{ik} x_{j'}$$

$$\leq\; \overline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \overline{a}_{jj'}^{ik} x_{j'}$$

Otherwise, if $\mathrm{Var}^i \neq k$, then Equations B.24-25 become

$$x_j \;\geq\; \underline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \underline{a}_{jj'}^{ik} x_{j'} - \left(\overline{x}_j - \underline{x}_j\right)$$

$$\leq\; \overline{a}_j^{ik} + \sum_{\substack{j'=1 \\ j' \neq j}}^{n} \overline{a}_{jj'}^{ik} x_{j'} + \left(\overline{x}_j - \underline{x}_j\right)$$

which are automatically satisfied, given that $\underline{x}_j \leq x_j \leq \overline{x}_j$. For this second case, when $\beta_{\mathrm{Var}k}^{i} \neq 1$, Equations B.24-25 are said to be slack. This notion of slackness coincides with the discussion in Appendix B.3, albeit in a semi-discrete context.

Equations B.22-25 provide a particular structure which is often observed in scheduling problems [192], and which is amenable to specialized cutting algorithms that are superimposed onto the standard Branch-and-Cut approach [190, 193]. Since the early 2000's, these specialized Branch-and-Cut approaches have been hybridized with filtered search techniques, forming part of the Constraint Programming paradigm [93].

It is an ongoing area of research to develop specialized solution techniques for MILP's with categorical variables,. This area is especially lucrative, considering the applications in industrial scheduling [93, 192], and engineering design [194, 195].

B.6 Importance of Linear Fractional Programming

Linear fractional programming (LFP) is a well-known class of nonlinear mathematical programming, which is only marginally more complicated than linear programming, unless integer constraints are imposed [196]. Although LFP is nonlinear, it has a particular importance to MILP, as it can help manage the branching procedures for structured problems, e.g. in schedule optimization.

A (continuous) linear fractional program has the general form,

$$\begin{aligned} \max_{x_j} f = \quad & \frac{c_\circ + \sum_{j=1}^{n} c_j x_j}{d_\circ + \sum_{j=1}^{n} d_j x_j} \\ \text{such that} \quad & \sum_{j=1}^{n} a_{ij}^{\text{lt}} x_j \leq b_i \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\ & \sum_{j=1}^{n} a_{ij}^{\text{gt}} x_j \geq b_i^{\text{gt}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \\ & \sum_{j=1}^{n} a_{ij}^{\text{eq}} x_j = b_i^{\text{eq}} \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{eq}}\} \end{aligned} \tag{6.49}$$

Without loss of generality, the objective denominator is presumed to be strictly positive, $d_\circ + \sum_{j=1}^{n} d_j x_j \geq \epsilon > 0$, for all feasible solutions; ϵ is an appropriately small positive number. Alternatively, if the denominator is strictly negative, then a the objective can be multiplied by $(\frac{-1}{-1})$ to transfer the negativity into the numerator. Lastly, if the denominator spans positive and negative values, then the objective function would be unbounded, tending toward infinity as the denominator approaches zero from the positive direction; this special case is not of

practical interest.

In industrial contexts, a linear program (Equation B.1) may be to optimize production. The corresponding fractional program (Equation B.26) would be to optimize productivity, meaning the production per unit consumption of some scarce resource [196]; thus production is placed in the numerator, and consumption is placed in the denominator. For scheduling problems, Equation B.26 may be to maximize the rate of production; hence the denominator would be a duration of time. More generally, Equation B.26 represents the optimization of some measure of efficiency.

Using the Charnes-Cooper Transformation [196], Equation B.26 is converted into the following linear program,

$$\begin{aligned}
\max_{(r,\tilde{x}_j)} f = \quad & c_\circ r + \sum_{j=0}^{n} c_j \tilde{x}_j \\
\text{such that} \quad & \sum_{j=1}^{n} a_{ij}^{\text{lt}} \tilde{x}_j \leq b_i r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{gt}} \tilde{x}_j \geq b_i^{\text{gt}} r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{eq}} \tilde{x}_j = b_i^{\text{eq}} r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{eq}}\} \\
& d_\circ r + \textstyle\sum_{j=1}^{n} d_j \tilde{x}_j = 1
\end{aligned} \tag{6.50}$$

such that

$$x_j = r\tilde{x}_j \tag{6.51}$$

for all $j \in \{1, 2, \ldots, n\}$. Thus Equation B.27 can be solved via the techniques presented in Appendices B.1 and B.2, and the final solution is subsequently obtained from Equation B.28.

Complications arise when integral constraints are introduced into Equation B.26, as the

integrality conditions of Equation B.16 are convolved with Equation B.28,

$$\begin{aligned}
\max_{(r,\tilde{x}_j)} f = \quad & c_\circ r + \sum_{j=0}^{n} c_j \tilde{x}_j \\
\text{such that} \quad & \sum_{j=1}^{n} a_{ij}^{\text{lt}} \tilde{x}_j \leq b_i r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{lt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{gt}} \tilde{x}_j \geq b_i^{\text{gt}} r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{gt}}\} \\
& \sum_{j=1}^{n} a_{ij}^{\text{eq}} \tilde{x}_j = b_i^{\text{eq}} r \qquad \text{for all } i \in \{1, 2, \ldots, m^{\text{eq}}\} \\
& d_\circ r + \textstyle\sum_{j=1}^{n} d_j \tilde{x}_j = 1 \\
& r\tilde{x}_j \in \{rz |\ z \in \mathcal{Z}\} \qquad \text{for all } j \in \{1, 2, \ldots, n_{\text{Int}}\}
\end{aligned} \tag{6.52}$$

Equation B.29 is the application of the Charnes-Cooper Transformation onto the general form of a mixed integer linear fractional program (MILFP).

In an attempt to adapt the branching procedure of Appendix B.4 to an MILFP, Equations B.17-18 become bilinear equations,

$$r\tilde{x}_{(j_k)} \leq \lfloor r^k \tilde{x}^k_{(j_k)} \rfloor$$

and

$$r\tilde{x}_{(j_k)} \geq \lceil r^k \tilde{x}^k_{(j_k)} \rceil$$

in which $(r^k, \tilde{x}_1^k, \tilde{x}_2^k, \ldots, \tilde{x}_n^k)$ is the solution obtained at the k^{th} branching node. These bilinear equations are not supported by the linear duality structure presented Appendix B.3. Thus, standard MILP solvers are not directly able to accommodate Equation B.29.

The most popular techniques for solving MILFP are based on Dinkelbach's algorithm [196], which reduces an MILFP into a sequence of MILP problems. This approach is especially noted for its uses in schedule optimization [197, 198]. For instance, it can incorporated into a Beam Search procedure [191], to identify which nodes are likely to be the most productive.

Hybridization of MILFP and MILP is yet another area of research that relates to scheduling and systems management. The outcomes from these efforts are not well-known outside of the optimization community, which suggests a tremendous potential.

APPENDIX C

AMPL FILES USED FOR THE SC-PSC PROBLEM

C.1 *mod* File

The SC-PSC Problem has been implemented in AMPL [77]. Thus the following AMPL code is a transliteration of the MILP model presented in Chapter 4 and Section 5.1. The order of the sections, subsections and constraints is respected. The sets, parameters and variables are introduced as necessary.

```
#=============================================================================================
#=============================================================================================
#=============================================================================================
#============================= SECTION 4.1 : GANTT STRUCTURE ==============================
#:::::::::::::::::::::::::::: Subsection 4.1.1 : Assignments ::::::::::::::::::::::::::::::
#Sets and Related Parameters
#--------------------------
set CLASSES;
param n{CLASSES} >= 0 integer, default 1;
param n_Asgn_max {CLASSES} >= 0 integer;
set ASSIGNMENTS = {l1 in CLASSES, l2 in 1..n[l1], l3 in 0..n_Asgn_max[l1]};
set ASSIGNMENTS_0 = {(l1,l2,l3) in ASSIGNMENTS: l3 = 0};
set ASSIGNMENTS_CURRENT = ASSIGNMENTS diff ASSIGNMENTS_0;
set TYPES {CLASSES};
#Check that object classes do not share assignment types
    check {i in CLASSES, ip in CLASSES:i <> ip}:
        card(TYPES[i] inter TYPES[ip]) = 0;
param PSCSymbol symbolic in CLASSES;
set ASSIGNMENTS_PSC = {(PSCSymbol,l2,l3) in ASSIGNMENTS};
set ASSIGNMENTS_0_PSC = {(PSCSymbol,l2,l3) in ASSIGNMENTS: l3 = 0};
set ASSIGNMENTS_CURRENT_PSC = ASSIGNMENTS_PSC diff ASSIGNMENTS_0_PSC;
set TYPES_PSC = TYPES[PSCSymbol];
set TYPES_PSC_Empty within TYPES_PSC;
set TYPES_PSC_minus {TYPES_PSC} within TYPES_PSC default {};

#Variables and Related Parameters (Including Initial Conditions)
#---------------------------------
param t_Begin default 0;
param t_End >= t_Begin;
param t_max >= t_End, default t_End;
param d_max >=0, <= (t_max - t_Begin), default (t_max - t_Begin);
var d {ASSIGNMENTS_CURRENT} >=0, <= d_max;
var t {(l1,l2,l3) in ASSIGNMENTS} >= t_Begin - (if (l3 = 0) then d_max), <= t_max;
    param t_0 {(l1,l2,l3) in ASSIGNMENTS_0} default t_Begin;
    subject to InitialConditions_t{(l1,l2,l3) in ASSIGNMENTS_0}:
```

```
        t[l1,l2,l3] = t_0[l1,l2,l3];
var B_Type {k in (union {i in CLASSES} TYPES[i]), (l1,l2,l3) in ASSIGNMENTS:
    k in TYPES[l1]} binary;
        param PreviousTypePSC symbolic in TYPES_PSC;
        param B_Type_0 {k in (union {i in CLASSES} TYPES[i]),
            (l1,l2,l3) in ASSIGNMENTS_0: k in TYPES[l1]}
        default if l1 = PSCSymbol then
                (if k = PreviousTypePSC then 1 else 0)
                else 1;
    subject to InitialConditions_B_Type {(l1,l2,l3) in ASSIGNMENTS_0, k in TYPES[l1]}:
        B_Type[k,l1,l2,l3] = B_Type_0[k,l1,l2,l3];

#Constraints
#-----------
subject to Chp4_Eq001_CurrentAssignmentsBeginInCurrentSchedule
    {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
        t[l1,l2,l3] - d[l1,l2,l3] >= t_Begin;
subject to Chp4_Eq002_AssignmentsBeginAfterPredecessorsEnd
    {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
        t[l1,l2,l3] - d[l1,l2,l3] >= t[l1,l2,l3-1];
subject to Chp4_Eq003_AssignmentTypeSelection {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
    sum {k in TYPES[l1]} B_Type[k,l1,l2,l3] <= 1;
subject to Chp4_Eq004_ZeroDurationForUnassigned {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
    d[l1,l2,l3] <= d_max*(sum {k in TYPES[l1]} B_Type[k,l1,l2,l3]);
subject to Chp4_Eq005_DeterminedPredecessor {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
    sum {k in TYPES[l1]} B_Type[k,l1,l2,l3] <=
        sum {k in TYPES[l1]} B_Type[k,l1,l2,l3-1];
subject to Chp4_Eq006_CarryIntoNextSchedule {(l1,l2,l3) in ASSIGNMENTS_CURRENT}:
    t[l1,l2,l3-1] <=
        t_End + (t_max - t_End)*(1 - sum {k in TYPES[l1]} B_Type[k,l1,l2,l3]);
#subject to Chp4_Eq007_TimeBoundForl3: (implemented in the declaration of t)


#:::::::::::::::::::::::: Subsection 4.1.2 : Dependencies ::::::::::::::::::::::::::::
#Sets
#----
set DEPENDENCIES {i in CLASSES, k in TYPES[i]} within
    {ip in CLASSES, kp in TYPES[ip], np in 1..n[ip]:
        ip <> i or kp <> k} default {};

#Variables
#---------
var B_Supp {k in (union {l1 in CLASSES} TYPES[l1]), (i,l2,l3) in ASSIGNMENTS_CURRENT,
    kp in (union {l1 in CLASSES} TYPES[l1]), (ip,l2p,l3p) in ASSIGNMENTS_CURRENT :
                k in TYPES[i] and exists {(ipp,kpp,npp) in DEPENDENCIES[i,k]}
                (ipp = ip and kpp = kp)}
    binary;

#Constraints
#-----------
subject to Chp4_Eq008_DependencyClause {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT, (ip,kp,np) in DEPENDENCIES[i,k]}:
        sum{(ip,l2p,l3p) in ASSIGNMENTS_CURRENT}
            B_Supp[k,i,l2,l3,kp,ip,l2p,l3p] = np*B_Type[k,i,l2,l3];
```

```
subject to Chp4_Eq009_TypeSupportConsistency {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT,
    (ip,kp,np) in DEPENDENCIES[i,k], (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        2*B_Supp[k,i,l2,l3,kp,ip,l2p,l3p] <=
            B_Type[k,i,l2,l3] + B_Type[kp,ip,l2p,l3p];
subject to Chp4_Eq010_SupportNoMoreThanOneAssignment {i in CLASSES, k in TYPES[i],
    (ip,kp,np) in DEPENDENCIES[i,k], (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        B_Type[kp,ip,l2p,l3p] =
            sum{(i,l2,l3) in ASSIGNMENTS_CURRENT} B_Supp[k,i,l2,l3,kp,ip,l2p,l3p];
subject to Chp4_Eq011_SimultaneousDuration {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT, (ip,kp,np) in DEPENDENCIES[i,k],
    (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        d[i,l2,l3] >= d[ip,l2p,l3p] - d_max*(1 - B_Supp[k,i,l2,l3,kp,ip,l2p,l3p]);
subject to Chp4_Eq012_SimultaneousDuration {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT, (ip,kp,np) in DEPENDENCIES[i,k],
    (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        d[i,l2,l3] <= d[ip,l2p,l3p] + d_max*(1 - B_Supp[k,i,l2,l3,kp,ip,l2p,l3p]);
subject to Chp4_Eq013_SimultaneousCompletionTime {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT, (ip,kp,np) in DEPENDENCIES[i,k],
    (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        t[i,l2,l3] >= t[ip,l2p,l3p] - t_max*(1 - B_Supp[k,i,l2,l3,kp,ip,l2p,l3p]);
subject to Chp4_Eq014_SimultaneousCompletionTime {i in CLASSES, k in TYPES[i],
    (i,l2,l3) in ASSIGNMENTS_CURRENT, (ip,kp,np) in DEPENDENCIES[i,k],
    (ip,l2p,l3p) in ASSIGNMENTS_CURRENT}:
        t[i,l2,l3] <= t[ip,l2p,l3p] + t_max*(1 - B_Supp[k,i,l2,l3,kp,ip,l2p,l3p]);


#============== SECTION 4.2 : PEIRCE-SMITH CONVERTERS AS STATE-MACHINES ===============
#:::::::::::::::::::: Subsections 4.2.1 : States and Transitions ::::::::::::::::::::::
#Sets and Related Parameters
#---------------------------
set STREAMS_NGFeed;
set ELEMENTS;
set SPECIES_Prod;
set STREAMS_Prod;
    param CMatteSymbol symbolic in STREAMS_Prod;
    param BlisterSymbol symbolic in STREAMS_Prod diff {CMatteSymbol};
    param SlagSymbol symbolic in STREAMS_Prod diff {CMatteSymbol, BlisterSymbol};
    param OffgasSymbol symbolic in STREAMS_Prod diff
                    {CMatteSymbol, BlisterSymbol, SlagSymbol};
        check: #Check that STREAMS_Prod = {CMatte, Blister, Slag, Offgas}
            card(STREAMS_Prod) = 4;
set STREAMS_NGProd = STREAMS_Prod diff {OffgasSymbol};
set SPECIES {STREAMS_Prod} within SPECIES_Prod;
    #Check that each product species only reports to one product stream
    check {k in STREAMS_Prod, kp in STREAMS_Prod: k <> kp}:
        card(SPECIES[k] inter SPECIES[kp]) = 0;
    #Check that each product species is accounted for in at least one product
    check:
        card(union {k in STREAMS_Prod} SPECIES[k]) = card(SPECIES_Prod);
set SPECIES_Offgas = SPECIES[OffgasSymbol];
set SPECIES_NGProd = SPECIES_Prod diff SPECIES_Offgas;
set FLOWS_NG;
set FLOWS_Ch within FLOWS_NG;
```

```
set FLOWS_NGBlow within FLOWS_NG diff FLOWS_Ch;
set FLOWS_NGFeed = FLOWS_Ch union FLOWS_NGBlow;
set FLOWS_DCh = FLOWS_NG diff FLOWS_NGFeed;
set FLOWS {STREAMS_NGFeed union STREAMS_NGProd} within FLOWS_NG default {};
#Check that each nongaseous mechanism draws on only one nongaseous stream
    check {k in STREAMS_NGFeed union STREAMS_NGProd,
        kp in STREAMS_NGFeed union STREAMS_NGProd: k <> kp}:
            card(FLOWS[k] inter FLOWS[kp]) = 0;
#Check that nongaseous feed flows map to nongaseous feed streams
    check {k in STREAMS_NGFeed}:
        if FLOWS[k] within FLOWS_NGFeed then 1 else 0 = 1;
#Check that nongaseous product flows map to discharge streams
    check {k in STREAMS_NGProd}:
        if FLOWS[k] within FLOWS_DCh then 1 else 0 = 1;
set FLOWS_M_Ch within FLOWS_Ch;
set FLOWS_SM_Ch within FLOWS_Ch diff FLOWS_M_Ch;
set FLOWS_UM_Ch = FLOWS_M_Ch diff (FLOWS_M_Ch union FLOWS_SM_Ch);
set FLOWS_M_DCh within FLOWS_DCh;
set FLOWS_SM_DCh within FLOWS_DCh diff FLOWS_M_DCh;
set FLOWS_UM_DCh = FLOWS_M_DCh diff (FLOWS_M_DCh union FLOWS_SM_DCh);
set FLOWS_M = FLOWS_M_Ch union FLOWS_M_DCh;
set FLOWS_SM = FLOWS_SM_Ch union FLOWS_SM_DCh;
set FLOWS_MSM_Ch = FLOWS_M_Ch union FLOWS_SM_Ch;
set FLOWS_MSM_DCh = FLOWS_M_DCh union FLOWS_SM_DCh;
set FLOWS_MSM = FLOWS_MSM_Ch union FLOWS_MSM_DCh;

#Variables and Related Parameters (Including Initial Conditions)
#--------------------------------
param v_PSC_max {1..n[PSCSymbol]} >= 0;
param rho{STREAMS_NGFeed union SPECIES_NGProd} > 0;
param m_max {STREAMS_NGFeed union ELEMENTS union SPECIES_Prod union STREAMS_NGProd}
    default 0;
var m_Ret {k in STREAMS_NGFeed, ASSIGNMENTS_PSC} >= 0, <= m_max[k];
    param m_0_Ret {STREAMS_NGFeed, 1..n[PSCSymbol]} >=0, default 0;
    subject to InitialConditions_m_Ret {k in STREAMS_NGFeed,
        (l2,l3) in ASSIGNMENTS_0_PSC}:
m_Ret[k,l2,l3] = m_0_Ret[k,l2];
var m_RetProd {j in SPECIES_NGProd, ASSIGNMENTS_PSC} >= 0, <= m_max[j];
    param m_0_RetProd {SPECIES_NGProd, 1..n[PSCSymbol]} >=0, default 0;
    subject to InitialConditions_m_RetProd {j in SPECIES_NGProd,
        (l2,l3) in ASSIGNMENTS_0_PSC}:
            m_RetProd[j,l2,l3] = m_0_RetProd[j,l2];
param h_min {1..n[PSCSymbol]};
param h_max {1..n[PSCSymbol]};
var h_Ret {(l2,l3) in ASSIGNMENTS_PSC} >= h_min[l2], <= h_max[l2];
    param h_0_Ret {1..n[PSCSymbol]} default 0;
    subject to InitialConditions_h_Ret {(l2,l3) in ASSIGNMENTS_0_PSC}:
        h_Ret[l2,l3] = h_0_Ret[l2];
param d_Int_min {0..7} default 0;
param d_Int_max {i in 0..7} default if i = 0 then (t_End -t_Begin) else d_max;
var d_Int {i in 0..7, ASSIGNMENTS_CURRENT_PSC} >= d_Int_min[i], <= d_Int_max[i];
param v_min {FLOWS_NG} >= 0, default 0;
param v_max {j in FLOWS_NG} >= v_min[j],
    default max{jp in 1..n[PSCSymbol]} v_PSC_max[jp];
```

```
var v {j in FLOWS_NG, ASSIGNMENTS_CURRENT_PSC} >= v_min[j],  <= v_max[j];
param u_min {FLOWS_MSM} >= 0, default 0;
param u_max {j in FLOWS_MSM} >= u_min[j];
var u {j in FLOWS_MSM, ASSIGNMENTS_CURRENT_PSC} integer, >= u_min[j],  <= u_max[j];
param d_IntType_min {i in 0..7, TYPES_PSC} >= d_Int_min[i], default d_Int_min[i];
param d_IntType_max {i in 0..7, k in TYPES_PSC} >= d_IntType_min[i,k], <= d_Int_max[i],
    default d_IntType_min[i,k];
param v_Type_min {j in FLOWS_NG, TYPES_PSC} >= v_min[j], default v_min[j];
param v_Type_max {j in FLOWS_NG, k in TYPES_PSC} >= v_Type_min[j,k],
    <= v_max[j], default v_Type_min[j,k];
param u_Type_min {j in FLOWS_MSM, TYPES_PSC} >= u_min[j], default u_min[j];
param u_Type_max {j in FLOWS_MSM, k in TYPES_PSC} >= u_Type_min[j,k], <= u_max[j],
    default u_Type_min[j,k];
param v_u {FLOWS_M} >= 0;
param v_u_max {FLOWS_SM} >= 0;

#Constraints
#-----------
subject to Chp4_Eq015_DecomposePSCTransitionsIntoIntervals
    {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        d[PSCSymbol,l2,l3] =  sum {i in 1..7} d_Int[i,l2,l3];
subject to Chp4_Eq016_IntertransitionTime {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Int[0,l2,l3] = t[PSCSymbol,l2,l3] - d[PSCSymbol,l2,l3] - t[PSCSymbol,l2,l3 - 1];
subject to Chp4_Eq017_EmptyConverter {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    sum {k in STREAMS_NGFeed} m_Ret[k,l2,l3]/rho[k]
    + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3]/rho[j]  <=
            v_PSC_max[l2]*(1 - sum{k in TYPES_PSC_Empty} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq018_EmptyConverter {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Ret[l2,l3] >=
        h_min[l2]*(1 - sum{k in TYPES_PSC_Empty} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq019_EmptyConverter {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Ret[l2,l3] <=
        h_max[l2]*(1 - sum{k in TYPES_PSC_Empty} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq020_MechanisticPreparedness {k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        B_Type[k,PSCSymbol,l2,l3] <=
            sum{kp in TYPES_PSC_minus[k]} B_Type[kp,PSCSymbol,l2,l3-1];
subject to Chp4_Eq021_TypeBasedBounds {i in 0..7, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Int[i,l2,l3] >= sum{k in TYPES_PSC} d_IntType_min[i,k]*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq022_TypeBasedBounds {i in 0..7, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Int[i,l2,l3] <= d_Int_max[i]
        - sum{k in TYPES_PSC}
                (d_Int_max[i] - d_IntType_max[i,k])*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq023_TypeBasedBounds {i in 0..7, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Int[i,l2,l3] <= d_Int_max[i]*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq024_TypeBasedBounds {j in FLOWS_NG,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        v[j,l2,l3] >= sum{k in TYPES_PSC} v_Type_min[j,k]*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq025_TypeBasedBounds {j in FLOWS_NG,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        v[j,l2,l3] <= v_max[j]
                - sum{k in TYPES_PSC}
                    (v_max[j] - v_Type_max[j,k])*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq026_TypeBasedBounds {j in FLOWS_NG,
```

```
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        v[j,l2,l3] <= v_max[j]*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq027_TypeBasedBounds {j in FLOWS_MSM,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        u[j,l2,l3] >= sum{k in TYPES_PSC} u_Type_min[j,k]*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq028_TypeBasedBounds {j in FLOWS_MSM,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        u[j,l2,l3] <= u_max[j]
                - sum{k in TYPES_PSC}
                    (u_max[j] - u_Type_max[j,k])*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq029_TypeBasedBounds {j in FLOWS_MSM,
(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        u[j,l2,l3] <= u_max[j]*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq030_FlowVolumeModulation {j in FLOWS_M,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        v[j,l2,l3] = v_u[j]*u[j,l2,l3];
subject to Chp4_Eq031_FlowVolumeSemiModulation {j in FLOWS_SM,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        v[j,l2,l3] <= v_u_max[j]*u[j,l2,l3];


#:::::::::::::::::::::::: Subsection 4.2.2 : Converting Actions ::::::::::::::::::::::::
#Variables and Related Parameters
#--------------------------------
var d_Ch {(l2,l3) in ASSIGNMENTS_CURRENT_PSC};
    subject to d2Interpretation {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        d_Ch[l2,l3] = d_Int[2,l2,l3];
var d_Blow {(l2,l3) in ASSIGNMENTS_CURRENT_PSC};
    subject to d4Interpretation {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        d_Blow[l2,l3] = d_Int[4,l2,l3];
var d_DCh {(l2,l3) in ASSIGNMENTS_CURRENT_PSC};
    subject to d6Interpretation {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        d_DCh[l2,l3] = d_Int[6,l2,l3];
#Check that we use the lowest possible indices,
# while respecting (d_Ch,d_Blow,d_DCh) = (d_2,d_4,d_6)
    check {i in 1..3, k in TYPES_PSC}:
        if (d_IntType_max[2*i+1,k] = 0) then 3 >=
            if (d_IntType_max[2*i,k] = 0) then 1 +
                if (d_IntType_min[2*i,k] = 0) then 1 +
                if (d_IntType_min[2*i+1,k] = 0) then 1;
param d_v {FLOWS_NG} >= 0, default 0;
param d_u {FLOWS_MSM} >= 0, default 0;
param d_Ch_Type {TYPES_PSC} >= 0, default 0;
param d_DCh_Type {TYPES_PSC} >= 0, default 0;
param d_Blow_Type {TYPES_PSC} >= 0, default 0;

#Constraints
#-----------
subject to Chp4_Eq032_ChargingTime {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Ch[l2,l3] = sum{j in FLOWS_Ch} d_v[j]*v[j,l2,l3]
                + sum{j in FLOWS_MSM_Ch} d_u[j]*u[j,l2,l3]
                + sum{k in TYPES_PSC} d_Ch_Type[k]*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq033_DischargingTime {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_DCh[l2,l3] = sum{j in FLOWS_DCh} d_v[j]*v[j,l2,l3]
```

```
                + sum{j in FLOWS_MSM_DCh} d_u[j]*u[j,l2,l3]
                + sum{k in TYPES_PSC} d_DCh_Type[k]*B_Type[k,PSCSymbol,l2,l3];
subject to Chp4_Eq034_BlowingFluxTimeBound {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_Blow[l2,l3] >= sum{j in FLOWS_NGBlow} d_v[j]*v[j,l2,l3]
                + sum{k in TYPES_PSC} d_Blow_Type[k]*B_Type[k,PSCSymbol,l2,l3];


#======================== SECTION 4.3 : INTERMEDIATE COMPUTATIONS ======================
#:::::::::::::::::::: Subsections 4.3.1 : Intermediate Variables ::::::::::::::::::::::
#Variables and Related Parameters (Including Initial Conditions)
#--------------------------------
var m {k in STREAMS_NGFeed, (l2,l3) in ASSIGNMENTS_CURRENT_PSC} >= 0, <= m_max[k];
var m_Prod {i in ELEMENTS union SPECIES_Prod, (l2,l3) in ASSIGNMENTS_PSC} >= 0,
    <= (if l3 > 0 then m_max[i] else 0);
var h_Ch {(l2,l3) in ASSIGNMENTS_PSC} >= h_min[l2], <= h_max[l2];
var h_NGBlow {(l2,l3) in ASSIGNMENTS_PSC} >= h_min[l2], <= h_max[l2];
param m_Blast_max {ELEMENTS} >= 0, default 0;
var m_Blast {i in ELEMENTS, (l2,l3) in ASSIGNMENTS_PSC} >= 0,
    <= (if l3 > 0 then m_Blast_max[i] else 0);
param h_Blast_min default 0;
param h_Blast_max default 0;
var h_Blast {(l2,l3) in ASSIGNMENTS_PSC} >= (if l3 > 0 then h_Blast_min else 0),
    <= (if l3 > 0 then h_Blast_max else 0);
param h_Offgas_min;
param h_Offgas_max;
var h_Offgas {(l2,l3) in ASSIGNMENTS_PSC} >= h_Offgas_min, <= h_Offgas_max;
param h_DCh_min;
param h_DCh_max;
var h_DCh {(l2,l3) in ASSIGNMENTS_PSC} >= h_DCh_min, <= h_DCh_max;
param h_Env_max {0..7} >= 0, default 0;
var h_Env {i in 0..7, (l2,l3) in ASSIGNMENTS_PSC} >= 0,
    <= (if l3 > 0 then h_Env_max[i] else 0);
param w {ELEMENTS, STREAMS_NGFeed union SPECIES_Prod} >= 0, default 0;
param w_Feed_H {FLOWS_NGFeed};

#Constraints
#-----------
subject to Chp4_Eq035_FeedMass {k in STREAMS_NGFeed, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    m[k,l2,l3] = m_Ret[k,l2,l3-1] + sum {j in FLOWS[k]} rho[k]*v[j,l2,l3];
subject to Chp4_Eq036_ElementalProductMass {i in ELEMENTS,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Prod[i,l2,l3] = sum {j in SPECIES_NGProd} w[i,j]*m_RetProd[j,l2,l3-1]
            + sum {k in STREAMS_NGFeed} w[i,k]*(m[k,l2,l3] - m_Ret[k,l2,l3])
            + m_Blast[i,l2,l3];
subject to Chp4_Eq037_ChargeHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Ch[l2,l3] = sum {k in STREAMS_NGFeed, j in FLOWS[k] diff FLOWS_NGBlow}
                    w_Feed_H[j]*rho[k]*v[j,l2,l3];
subject to Chp4_Eq038_NongaseousBlowHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_NGBlow[l2,l3] = sum {k in STREAMS_NGFeed, j in FLOWS[k] inter FLOWS_NGBlow}
                    w_Feed_H[j]*rho[k]*v[j,l2,l3];


#:::::::::::::::::::::: Subsection 4.3.2 : Blast Elemental Masses ::::::::::::::::::::::
#Parameters
```

```
#----------
param m_Blast_dot {ELEMENTS, TYPES_PSC} >= 0, default 0;

#Constraints
#-----------
subject to Chp4_Eq039_BlastMass {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Blast[i,l2,l3] >= m_Blast_dot[i,k]*d_Blow[l2,l3]
                        - m_Blast_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq040_BlastMass {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Blast[i,l2,l3] <= m_Blast_dot[i,k]*d_Blow[l2,l3]
                        + m_Blast_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq041_UndeterminedBlastMass {i in ELEMENTS,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Blast[i,l2,l3] <=
            m_Blast_max[i]*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);


#:::::::::::::::::::::: Subsection 4.3.3 : Product Species Masses ::::::::::::::::::::::
#Sets and Related Parameters
#---------------------------
set REGIMES;
set SPECIES_RgProd within SPECIES_NGProd;
set REGIMES_spec {SPECIES_RgProd} within REGIMES;
set TYPES_PSC_Prod within TYPES_PSC;
param OXYGEN_EFFICIENCY in [0,1];
param FERROSLAG_RATIO >= 0;
param epsilon_mProd default 0.0001;
param OSymbol symbolic in ELEMENTS;
param O2Symbol symbolic in SPECIES_Offgas;
param Fe2SiO4Symbol symbolic in SPECIES[SlagSymbol];
param Fe3O4Symbol symbolic in SPECIES[SlagSymbol];

#Variables
#---------
var B_Rg {REGIMES, ASSIGNMENTS_CURRENT_PSC} binary;

#Constraints
#-----------
subject to Chp4_Eq042_SpeciesRegimeElimination {j in SPECIES_RgProd,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Prod[j,l2,l3] <= m_max[j]*(sum {k in REGIMES_spec[j]} B_Rg[k,l2,l3]);
subject to Chp4_Eq043_GlobalElementSpeciesMassBalance {i in ELEMENTS,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{j in SPECIES_Prod} w[i,j]*m_Prod[j,l2,l3] = m_Prod[i,l2,l3];
subject to Chp4_Eq044_RegimeSelection {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    sum {k in REGIMES} B_Rg[k,l2,l3] = card(REGIMES)
        - (card(REGIMES)- 1)*(sum {k in TYPES_PSC_Prod} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq045_SpeciesRegimeElimination {j in SPECIES_Prod,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Prod[j,l2,l3] <= m_max[j] - (m_max[j] - epsilon_mProd)*
            (1 - sum {k in TYPES_PSC_Prod} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq046_OxygenEfficiency {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
```

```
    m_Prod[O2Symbol,l2,l3] = (1-OXYGEN_EFFICIENCY)*m_Blast[OSymbol,l2,l3];
subject to Chp4_Eq047_FerroSlagRatio {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    FERROSLAG_RATIO*m_Prod[Fe3O4Symbol,l2,l3] = 0.70514*m_Prod[Fe2SiO4Symbol,l2,l3];


#:::::::::::::::::::::::::: Subsection 4.3.4 : Blast Heat ::::::::::::::::::::::::::::
#Parameters
#----------
param h_Blast_dot {TYPES_PSC} default 0;

#Constraints
#-----------
subject to Chp4_Eq048_BlastHeat {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Blast[l2,l3] >= h_Blast_dot[k]*d_Blow[l2,l3]
                - (h_Blast_max - h_Blast_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq049_BlastHeat {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Blast[l2,l3] <= h_Blast_dot[k]*d_Blow[l2,l3]
                + (h_Blast_max - h_Blast_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq050_UndeterminedBlastHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Blast[l2,l3] >= h_Blast_min*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq051_UndeterminedBlastHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Blast[l2,l3] <= h_Blast_max*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);


#:::::::::::::::::::::::::: Subsection 4.3.5 : Offgas Heat ::::::::::::::::::::::::::::
#Parameters
#----------
param w_Offgas_H {j in SPECIES_Offgas, TYPES_PSC};

#Constraints
#-----------
subject to Chp4_Eq052_OffgasHeat {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Offgas[l2,l3] >= sum {j in SPECIES_Offgas} w_Offgas_H[j,k]*m_Prod[j,l2,l3]
                - (h_Offgas_max - h_Offgas_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq053_OffgasHeat {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Offgas[l2,l3] <= sum {j in SPECIES_Offgas} w_Offgas_H[j,k]*m_Prod[j,l2,l3]
                + (h_Offgas_max - h_Offgas_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq054_UndeterminedOffgasHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Offgas[l2,l3] >= h_Offgas_min*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq055_UndeterminedOffgasHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Offgas[l2,l3] <= h_Offgas_max*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);


#:::::::::::::::::::::::::: Subsection 4.3.6 : Discharge Heat ::::::::::::::::::::::::::
#Sets and Parameters
#-------------------
set TYPES_PSC_IDCh = {k in TYPES_PSC diff TYPES_PSC_Empty:
    exists {j in FLOWS_DCh} v_Type_max[j,k] > 0};
param w_DCh_H {STREAMS_NGFeed union SPECIES_NGProd, TYPES_PSC_IDCh};

#Constraints
#-----------
subject to Chp4_Eq056_DChHeat {k in TYPES_PSC_IDCh, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_DCh[l2,l3] >= sum {j in SPECIES_NGProd}
```

```
        w_DCh_H[j,k]*(m_Prod[j,l2,l3] - m_RetProd[j,l2,l3])
                - (h_DCh_max - h_DCh_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq057_DChHeat {k in TYPES_PSC_IDCh,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_DCh[l2,l3] <= sum {j in SPECIES_NGProd}
                w_DCh_H[j,k]*(m_Prod[j,l2,l3] - m_RetProd[j,l2,l3])
                + (h_DCh_max - h_DCh_min)*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq058_UndeterminedDChHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_DCh[l2,l3] >= h_DCh_min*(sum{k in TYPES_PSC_Empty union TYPES_PSC_IDCh}
                            B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq059_UndeterminedDChHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_DCh[l2,l3] <= h_DCh_max*(sum{k in TYPES_PSC_Empty union TYPES_PSC_IDCh}
                                B_Type[k,PSCSymbol,l2,l3]);


#:::::::::::::::::::: Subsection 4.3.7 : Environmental Heat Losses ::::::::::::::::::::
#Parameters
#----------
param h_EnvType {0..7, TYPES_PSC} >= 0, default 0;
param h_EnvType_dot {0..7, TYPES_PSC} >= 0, default 0;

#Constraints
#-----------
subject to Chp4_Eq060_EnvHeatLoss {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Env[0,l2,l3] >= h_EnvType[0,k]*B_Type[k,PSCSymbol,l2,l3-1]
                + h_EnvType_dot[0,k]*d_Int[0,l2,l3]
                - h_Env_max[0]*(1 - B_Type[k,PSCSymbol,l2,l3-1]);
subject to Chp4_Eq061_EnvHeatLoss {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Env[0,l2,l3] <= h_EnvType[0,k]*B_Type[k,PSCSymbol,l2,l3-1]
                + h_EnvType_dot[0,k]*d_Int[0,l2,l3]
                + h_Env_max[0]*(1 - B_Type[k,PSCSymbol,l2,l3-1]);
subject to Chp4_Eq062_EnvHeatLoss {i in 1..7,
    k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Env[i,l2,l3] >= h_EnvType[i,k]*B_Type[k,PSCSymbol,l2,l3]
                + h_EnvType_dot[i,k]*d_Int[i,l2,l3]
                - h_Env_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq063_EnvHeatLoss {i in 1..7,
    k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Env[i,l2,l3] <= h_EnvType[i,k]*B_Type[k,PSCSymbol,l2,l3]
                + h_EnvType_dot[i,k]*d_Int[i,l2,l3]
                + h_Env_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq064_UndeterminedEnvHeatLoss {i in 0..7,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Env[i,l2,l3] <=
            h_Env_max[i]*(sum{k in TYPES_PSC} B_Type[k,PSCSymbol,l2,l3]);


#======================== SECTION 4.4 : FORWARD COMPUTATIONS ==========================
#:::::::::::::::::::: Subsection 4.4.1 : Retained Feed Masses ::::::::::::::::::::::::
#Constraints
#-----------
subject to Chp4_Eq065_CompleteReaction {k in STREAMS_NGFeed,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] >= m[k,l2,l3]
```

```
            - m_max[k]*sum{kp in TYPES_PSC: d_IntType_max[4,kp] > 0}
                B_Type[kp,PSCSymbol,l2,l3];
subject to Chp4_Eq066_CompleteReaction {k in STREAMS_NGFeed,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] <= m[k,l2,l3]
            + m_max[k]*sum{kp in TYPES_PSC: d_IntType_max[4,kp] > 0}
                B_Type[kp,PSCSymbol,l2,l3];
subject to Chp4_Eq067_CompleteReaction {k in STREAMS_NGFeed,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] <= m_max[k]*(1 -
            sum{kp in TYPES_PSC: d_IntType_max[4,kp] > 0}
                B_Type[kp,PSCSymbol,l2,l3]);


#::::::::::::::::: Subsection 4.4.2 : Retained Product Species Masses :::::::::::::::::
#Constraints
#-----------
subject to Chp4_Eq068_CompleteDischarge {kp in STREAMS_NGProd, j in SPECIES[kp],
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_RetProd[j,l2,l3] >= m_Prod[j,l2,l3]
        - m_max[j]*(sum{k in TYPES_PSC: exists {jp in FLOWS[kp]} v_Type_max[jp,k] > 0}
                B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq069_CompleteDischarge {kp in STREAMS_NGProd, j in SPECIES[kp],
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_RetProd[j,l2,l3] <= m_Prod[j,l2,l3]
        + m_max[j]*(sum{k in TYPES_PSC: exists {jp in FLOWS[kp]} v_Type_max[jp,k] > 0}
                    B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq070_CompleteDischarge {kp in STREAMS_NGProd, j in SPECIES[kp],
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_RetProd[j,l2,l3] <= m_max[j]*(1 - sum{k in TYPES_PSC:
                        exists {jp in FLOWS[kp]} v_Type_max[jp,k] > 0}
                            B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq071_DischargeStreamVolume {k in STREAMS_NGProd,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{j in SPECIES[k]} (m_Prod[j,l2,l3] - m_RetProd[j,l2,l3])/rho[j] =
            sum{j in FLOWS[k]} v[j,l2,l3];


#:::::::::::::::::::::: Subsection 4.4.3 : Forward Heat Computation ::::::::::::::::::
#Constraints
#-----------
subject to Chp4_Eq072_RetainedHeat {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    h_Ret[l2,l3] = h_Ret[l2,l3-1] + h_Ch[l2,l3] + h_NGBlow[l2,l3] + h_Blast[l2,l3]
                - h_Offgas[l2,l3] - h_DCh[l2,l3] - sum{i in 0..7} h_Env[i,l2,l3];


#==================== SECTION 4.5 : FEASIBLE CONVERTER TRANSITIONS ======================
#::::::: Subsection 4.5.1 : Direct Transition Constraints in General Linear Form ::::::::
#Sets and Related Parameters
#---------------------------
set DIRECT_TRANSITION_CONSTRAINTS_PSC default {};
    param a_direct_mRet_ {DIRECT_TRANSITION_CONSTRAINTS_PSC, STREAMS_NGFeed} default 0;
    param a_direct_mRetProd_ {DIRECT_TRANSITION_CONSTRAINTS_PSC,
        SPECIES_NGProd} default 0;
```

```
    param a_direct_hRet_ {DIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_direct_BType_ {DIRECT_TRANSITION_CONSTRAINTS_PSC, TYPES_PSC} default 0;
    param a_direct_d {DIRECT_TRANSITION_CONSTRAINTS_PSC, 0..7};
    param a_direct_v {DIRECT_TRANSITION_CONSTRAINTS_PSC, FLOWS_NG} default 0;
    param a_direct_u {DIRECT_TRANSITION_CONSTRAINTS_PSC, FLOWS_MSM} default 0;
    param a_direct_BType {DIRECT_TRANSITION_CONSTRAINTS_PSC, TYPES_PSC} default 0;
    param b_direct {DIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;

#Constraints
#-----------
subject to Chp4_Eq073_DirectTransition {i in DIRECT_TRANSITION_CONSTRAINTS_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {k in STREAMS_NGFeed} a_direct_mRet_[i,k]*m_Ret[k,l2,l3-1]
        + sum {j in SPECIES_NGProd} a_direct_mRetProd_[i,j]*m_RetProd[j,l2,l3-1]
        + a_direct_hRet_[i]*h_Ret[l2,l3-1]
        + sum {k in TYPES_PSC} a_direct_BType_[i,k]*B_Type[k,PSCSymbol,l2,l3-1]
        + sum {ip in 0..7} a_direct_d[i,ip]*d_Int[ip,l2,l3]
        + sum {j in FLOWS_NG} a_direct_v[i,j]*v[j,l2,l3]
        + sum {j in FLOWS_MSM} a_direct_u[i,j]*u[j,l2,l3]
        + sum {k in TYPES_PSC} a_direct_BType[i,k]*B_Type[k,PSCSymbol,l2,l3]
        <= b_direct[i];


#:::::::::::::::::::: Subsection 4.5.2 : Bath Composition Constraints ::::::::::::::::::
#Parameters
#----------
param w_Bath_0_min {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd, TYPES_PSC}
    >= 0,
    <= if i in ELEMENTS then
        max {k in STREAMS_NGFeed union SPECIES_NGProd} w[i,k] else 1,
    default 0;
param w_Bath_0_max {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd,
    k in TYPES_PSC}
        >= w_Bath_0_min[i,k],
        <= if i in ELEMENTS then max {kp in STREAMS_NGFeed union SPECIES_NGProd}
            w[i,kp] else 1,
    default (if i in ELEMENTS then max {kp in STREAMS_NGFeed union SPECIES_NGProd}
            w[i,kp] else 1);
param w_Bath_Blow_min {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd,
    TYPES_PSC}
        >= 0,
        <= if i in ELEMENTS then
            max {k in STREAMS_NGFeed union SPECIES_NGProd} w[i,k] else 1,
        default 0;
param w_Bath_Blow_max {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd,
    k in TYPES_PSC}
        >= w_Bath_Blow_min[i,k],
        <= if i in ELEMENTS then
            max {kp in STREAMS_NGFeed union SPECIES_NGProd} w[i,kp] else 1,
        default (if i in ELEMENTS then
            max {kp in STREAMS_NGFeed union SPECIES_NGProd} w[i,kp] else 1);
param w_Bath_DCh_min {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd,
    TYPES_PSC}
        >= 0,
```

```
        <= if i in ELEMENTS then max {j in SPECIES_NGProd} w[i,j] else 1,
        default 0;
param w_Bath_DCh_max {i in ELEMENTS union STREAMS_NGFeed union STREAMS_NGProd,
    k in TYPES_PSC}
        >= w_Bath_DCh_min[i,k],
        <= if i in ELEMENTS then
            max {kp in STREAMS_NGFeed union SPECIES_NGProd} w[i,kp] else 1,
        default (if i in ELEMENTS then
            max {kp in STREAMS_NGFeed union SPECIES_NGProd} w[i,kp] else 1);
param w_strm_O_min {i in ELEMENTS union SPECIES_NGProd, k in STREAMS_NGProd, TYPES_PSC}
    >= if i in ELEMENTS then min {j in SPECIES[k]} w[i,j] else 0,
    <= if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1,
    default (if i in ELEMENTS then min {j in SPECIES[k]} w[i,j] else 0);
param w_strm_O_max {i in ELEMENTS union SPECIES_NGProd, k in STREAMS_NGProd,
    kp in TYPES_PSC}
        >= w_strm_O_min[i,k,kp],
        <= if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1,
        default (if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1);
param w_strm_Blow_min {i in ELEMENTS union SPECIES_NGProd, k in STREAMS_NGProd,
    TYPES_PSC}
        >= if i in ELEMENTS then min {j in SPECIES[k]} w[i,j] else 0,
        <= if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1,
    default (if i in ELEMENTS then min {j in SPECIES[k]} w[i,j] else 0);
param w_strm_Blow_max {i in ELEMENTS union SPECIES_NGProd, k in STREAMS_NGProd,
    kp in TYPES_PSC}
        >= w_strm_Blow_min[i,k,kp],
        <= if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1,
    default (if i in ELEMENTS then max {j in SPECIES[k]} w[i,j] else 1);
param m_strm_max {i in ELEMENTS, k in STREAMS_NGProd} >= 0,
    default max {j in SPECIES[k]} w[i,j]*m_max[j];

#Constraints
#-----------
subject to Chp4_Eq074_BathInitialElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3-1]
        + sum{j in SPECIES_NGProd} w[i,j]*m_RetProd[j,l2,l3-1] >=
                w_Bath_O_min[i,k]*(sum{kp in STREAMS_NGFeed} m_Ret[kp,l2,l3-1]
                    + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
                - m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq075_BathInitialElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3-1]
        + sum{j in SPECIES_NGProd} w[i,j]*m_RetProd[j,l2,l3-1] <=
                w_Bath_O_max[i,k]*(sum{kp in STREAMS_NGFeed} m_Ret[kp,l2,l3-1]
                    + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
                + m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq076_BathBlowElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3]
        + sum {j in SPECIES_NGProd} w[i,j]*m_Prod[j,l2,l3] >=
            w_Bath_Blow_min[i,k]*(sum {kp in STREAMS_NGFeed} m_Ret[kp,l2,l3]
                    + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
                    - m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
```

```
subject to Chp4_Eq077_BathBlowElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3]
        + sum {j in SPECIES_NGProd} w[i,j]*m_Prod[j,l2,l3] <=
            w_Bath_Blow_max[i,k]*(sum {kp in STREAMS_NGFeed} m_Ret[kp,l2,l3]
                + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
                + m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq078_BathDChElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3]
        + sum {j in SPECIES_NGProd} w[i,j]*m_RetProd[j,l2,l3] >=
            w_Bath_DCh_min[i,k]*(sum {kp in STREAMS_NGFeed} m_Ret[kp,l2,l3]
                + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
                    - m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq079_BathDChElementalWeightFraction {i in ELEMENTS, k in TYPES_PSC,
(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {kp in STREAMS_NGFeed} w[i,kp]*m_Ret[kp,l2,l3]
        + sum {j in SPECIES_NGProd} w[i,j]*m_RetProd[j,l2,l3] <=
            w_Bath_DCh_max[i,k]*(sum {kp in STREAMS_NGFeed} m_Ret[kp,l2,l3]
                + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
                + m_max[i]*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq080_BathInitialFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3-1] >= w_Bath_O_min[k,kp]*
            (sum{kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3-1]
            + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
            - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq081_BathInitialFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3-1] <= w_Bath_O_max[k,kp]*
            (sum{kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3-1]
                + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
                + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq082_BathInitialProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{j in SPECIES[k]} m_RetProd[j,l2,l3-1] >=
            w_Bath_O_min[k,kp]*(sum{kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3-1]
                + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
                - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq083_BathInitialProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum{j in SPECIES[k]} m_RetProd[j,l2,l3-1] <=
            w_Bath_O_max[k,kp]*(sum{kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3-1]
                + sum{j in SPECIES_NGProd} m_RetProd[j,l2,l3-1])
                + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq084_BathBlowFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] >= w_Bath_Blow_min[k,kp]*
            (sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
            + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
            - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq085_BathBlowFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] <=
```

```
            w_Bath_Blow_max[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
                + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
                + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq086_BathBlowProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC,(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} m_Prod[j,l2,l3] >=
                w_Bath_Blow_min[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
                + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
                - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq087_BathBlowProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC,(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} m_Prod[j,l2,l3] <=
                w_Bath_Blow_max[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
                + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3])
                + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq088_BathDChFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] >=
            w_Bath_DCh_min[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
            + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
            - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq089_BathDChFeedStreamWeightFraction {k in STREAMS_NGFeed,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Ret[k,l2,l3] <=
            w_Bath_DCh_max[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
            + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
            + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq090_BathDChProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} m_RetProd[j,l2,l3] >=
            w_Bath_DCh_min[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
                + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
                - m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq091_BathDChProdStreamWeightFraction {k in STREAMS_NGProd,
    kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} m_RetProd[j,l2,l3] <=
            w_Bath_DCh_max[k,kp]*(sum {kpp in STREAMS_NGFeed} m_Ret[kpp,l2,l3]
                + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3])
                + m_max[k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq092_StreamInitialElementalWeightFraction {i in ELEMENTS,
    k in STREAMS_NGProd, kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} w[i,j]*m_RetProd[j,l2,l3-1] >=
            w_strm_0_min[i,k,kp]*(sum {j in SPECIES[k]} m_RetProd[j,l2,l3-1])
            - m_strm_max[i,k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq093_StreamInitialElementalWeightFraction {i in ELEMENTS,
    k in STREAMS_NGProd, kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} w[i,j]*m_RetProd[j,l2,l3-1] <=
            w_strm_0_max[i,k,kp]*(sum {j in SPECIES[k]} m_RetProd[j,l2,l3-1])
                    + m_strm_max[i,k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq094_StreamBlowElementalWeightFraction {i in ELEMENTS,
    k in STREAMS_NGProd, kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} w[i,j]*m_Prod[j,l2,l3] >=
            w_strm_Blow_min[i,k,kp]*(sum {j in SPECIES[k]} m_Prod[j,l2,l3])
                    - m_strm_max[i,k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
```

```
subject to Chp4_Eq095_StreamBlowElementalWeightFraction {i in ELEMENTS,
    k in STREAMS_NGProd, kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {j in SPECIES[k]} w[i,j]*m_Prod[j,l2,l3] <=
            w_strm_Blow_max[i,k,kp]*(sum {j in SPECIES[k]} m_Prod[j,l2,l3])
                + m_strm_max[i,k]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq096_StreamInitialSpeciesWeightFraction {k in STREAMS_NGProd,
    j in SPECIES[k], kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_RetProd[j,l2,l3-1] >=
            w_strm_O_min[j,k,kp]*(sum {jp in SPECIES[k]} m_RetProd[jp,l2,l3-1])
                - m_max[j]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq097_StreamInitialSpeciesWeightFraction {k in STREAMS_NGProd,
    j in SPECIES[k], kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_RetProd[j,l2,l3-1] <=
            w_strm_O_max[j,k,kp]*(sum {jp in SPECIES[k]} m_RetProd[jp,l2,l3-1])
                + m_max[j]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq098_StreamBathSpeciesWeightFraction {k in STREAMS_NGProd,
    j in SPECIES[k], kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Prod[j,l2,l3] >=
        w_strm_Blow_min[j,k,kp]*(sum {jp in SPECIES[k]} m_Prod[jp,l2,l3])
            - m_max[j]*(1 - B_Type[kp,PSCSymbol,l2,l3]);
subject to Chp4_Eq099_StreamBathSpeciesWeightFraction {k in STREAMS_NGProd,
    j in SPECIES[k], kp in TYPES_PSC, (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        m_Prod[j,l2,l3] <=
            w_strm_Blow_max[j,k,kp]*(sum {jp in SPECIES[k]} m_Prod[jp,l2,l3])
                + m_max[j]*(1 - B_Type[kp,PSCSymbol,l2,l3]);


#:::::::::::::::::::::::: Subsection 4.5.3 : Volume Constraints ::::::::::::::::::::::::
#Parameters
#----------
param v_PSC_Ch_max {TYPES_PSC, j in 1..n[PSCSymbol]} >= 0, <= v_PSC_max[j],
    default v_PSC_max[j];
param v_PSC_Blow_max {TYPES_PSC, j in 1..n[PSCSymbol]} >= 0, <= v_PSC_max[j],
    default v_PSC_max[j];

#Constraints
#-----------
subject to Chp4_Eq100_ChargeVolumeBound {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    sum {k in STREAMS_NGFeed} m_Ret[k,l2,l3-1]/rho[k]
    + sum {j in SPECIES_NGProd} m_RetProd[j,l2,l3-1]/rho[j]
    + sum {j in FLOWS_Ch} v[j,l2,l3]
    <= v_PSC_max[l2]
        - sum {k in TYPES_PSC}
        (v_PSC_max[l2] - v_PSC_Ch_max[k,l2])*B_Type[k, PSCSymbol, l2, l3];
subject to Chp4_Eq101_BlowVolumeBound {(l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
    sum {k in STREAMS_NGFeed} m_Ret[k,l2,l3]/rho[k]
    + sum {j in SPECIES_NGProd} m_Prod[j,l2,l3]/rho[j]
    <= v_PSC_max[l2]
        - sum {k in TYPES_PSC}
        (v_PSC_max[l2] - v_PSC_Blow_max[k,l2])*B_Type[k, PSCSymbol, l2, l3];


#:::::::::::::::::::: Subsection 4.5.4 : Temperature Constraints ::::::::::::::::::::
#Parameters
```

```
#----------
param w_Blow_H_min {STREAMS_NGFeed union SPECIES_NGProd, TYPES_PSC};
param w_Blow_H_max {STREAMS_NGFeed union SPECIES_NGProd, TYPES_PSC};
param w_DCh_H_min {j in STREAMS_NGFeed union SPECIES_NGProd, k in TYPES_PSC} default
    if k in TYPES_PSC_IDCh then w_DCh_H[j,k] else w_Blow_H_min[j,k];
param w_DCh_H_max {j in STREAMS_NGFeed union SPECIES_NGProd, k in TYPES_PSC} default
    if k in TYPES_PSC_IDCh then w_DCh_H[j,k] else w_Blow_H_max[j,k];
#Check that the intermediate discharge temperatures are fixed
    check {j in STREAMS_NGFeed union SPECIES_NGProd, k in TYPES_PSC_IDCh}:
        if (w_DCh_H_min[j,k] = w_DCh_H_max[j,k] and w_DCh_H_max[j,k] = w_DCh_H[j,k])
            then 1 = 1;

#Constraints
#-----------
subject to Chp4_Eq102_BlowTemperatureBound {k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Ret[l2,l3-1] + h_Ch[l2,l3] - sum{i in 0..3} h_Env[i,l2,l3]
            >= sum {kp in STREAMS_NGFeed} w_Blow_H_min[kp,k]*m_Ret[kp,l2,l3-1]
            + sum {j in SPECIES_NGProd} w_Blow_H_min[j,k]*m_RetProd[j,l2,l3-1]
            + sum {kp in STREAMS_NGFeed, j in FLOWS[kp] diff FLOWS_NGBlow}
                w_Blow_H_min[kp,k]*rho[kp]*v[j,l2,l3]
            - (h_max[l2] - h_min[l2])*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq103_BlowTemperatureBound {k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Ret[l2,l3-1] + h_Ch[l2,l3] + h_NGBlow[l2,l3] + h_Blast[l2,l3]
        - h_Offgas[l2,l3] - sum{i in 0..4} h_Env[i,l2,l3]
        <= sum {kp in STREAMS_NGFeed} w_Blow_H_max[kp,k]*m_Ret[kp,l2,l3]
            + sum {j in SPECIES_NGProd} w_Blow_H_max[j,k]*m_Prod[j,l2,l3]
            + (h_max[l2] - h_min[l2])*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq104_DChTemperatureBound {k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Ret[l2,l3-1] + h_Ch[l2,l3] + h_NGBlow[l2,l3] + h_Blast[l2,l3]
        - h_Offgas[l2,l3] - sum{i in 0..5} h_Env[i,l2,l3]
        >= sum {kp in STREAMS_NGFeed} w_DCh_H_min[kp,k]*m_Ret[kp,l2,l3]
  + sum {j in SPECIES_NGProd} w_DCh_H_min[j,k]*m_Prod[j,l2,l3]
  - (h_max[l2] - h_min[l2])*(1 - B_Type[k,PSCSymbol,l2,l3]);
subject to Chp4_Eq105_DChTemperatureBound {k in TYPES_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        h_Ret[l2,l3-1] + h_Ch[l2,l3] + h_NGBlow[l2,l3] + h_Blast[l2,l3]
        - h_Offgas[l2,l3] - sum{i in 0..5} h_Env[i,l2,l3]
        <= sum {kp in STREAMS_NGFeed} w_DCh_H_max[kp,k]*m_Ret[kp,l2,l3]
            + sum {j in SPECIES_NGProd} w_DCh_H_max[j,k]*m_Prod[j,l2,l3]
            + (h_max[l2] - h_min[l2])*(1 - B_Type[k,PSCSymbol,l2,l3]);


#:::::: Subsection 4.5.5 : Indirect Transition Constraints in General Linear Form :::::::
#Sets and Related Parameters
#---------------------------
set INDIRECT_TRANSITION_CONSTRAINTS_PSC default {};
    param a_indirect_mRet_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, STREAMS_NGFeed}
        default 0;
    param a_indirect_mRetProd_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, SPECIES_NGProd}
        default 0;
    param a_indirect_hRet_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
```

```
    param a_indirect_BType_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, TYPES_PSC} default 0;
    param a_indirect_d {INDIRECT_TRANSITION_CONSTRAINTS_PSC, 0..7} default 0;
    param a_indirect_v {INDIRECT_TRANSITION_CONSTRAINTS_PSC, FLOWS_NG} default 0;
    param a_indirect_u {INDIRECT_TRANSITION_CONSTRAINTS_PSC, FLOWS_MSM} default 0;
    param a_indirect_BType {INDIRECT_TRANSITION_CONSTRAINTS_PSC, TYPES_PSC} default 0;
    param a_indirect_mBlast_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, ELEMENTS} default 0;
    param a_indirect_mProd_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, SPECIES_Prod}
        default 0;
    param a_indirect_hBlast_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hOffgas_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hDCh_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hEnv_ {INDIRECT_TRANSITION_CONSTRAINTS_PSC, 0..7} default 0;
    param a_indirect_mBlast {INDIRECT_TRANSITION_CONSTRAINTS_PSC, ELEMENTS} default 0;
    param a_indirect_mProd {INDIRECT_TRANSITION_CONSTRAINTS_PSC, SPECIES_Prod}
        default 0;
    param a_indirect_hBlast {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hOffgas {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hDCh {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;
    param a_indirect_hEnv {INDIRECT_TRANSITION_CONSTRAINTS_PSC, 0..7} default 0;
    param b_indirect {INDIRECT_TRANSITION_CONSTRAINTS_PSC} default 0;

#Constraints
#-----------
subject to Chp4_Eq106_IndirectTransition {i in INDIRECT_TRANSITION_CONSTRAINTS_PSC,
    (l2,l3) in ASSIGNMENTS_CURRENT_PSC}:
        sum {k in STREAMS_NGFeed} a_indirect_mRet_[i,k]*m_Ret[k,l2,l3-1]
    + sum {j in SPECIES_NGProd} a_indirect_mRetProd_[i,j]*m_RetProd[j,l2,l3-1]
    + a_indirect_hRet_[i]*h_Ret[l2,l3-1]
    + sum {k in TYPES_PSC} a_indirect_BType_[i,k]*B_Type[k,PSCSymbol,l2,l3-1]
    + sum {ip in 0..7} a_indirect_d[i,ip]*d_Int[ip,l2,l3]
    + sum {j in FLOWS_NG} a_indirect_v[i,j]*v[j,l2,l3]
    + sum {j in FLOWS_MSM} a_indirect_u[i,j]*u[j,l2,l3]
    + sum {k in TYPES_PSC} a_indirect_BType[i,k]*B_Type[k,PSCSymbol,l2,l3]
    + sum {ip in ELEMENTS} a_indirect_mBlast_[i,ip]*m_Blast[ip,l2,l3-1]
    + sum {j in SPECIES_Prod} a_indirect_mProd_[i,j]*m_Prod[j,l2,l3-1]
    + a_indirect_hBlast_[i]*h_Blast[l2,l3-1]
    + a_indirect_hOffgas_[i]*h_Offgas[l2,l3-1]
    + a_indirect_hDCh_[i]*h_DCh[l2,l3-1]
    + sum {ip in 0..7} a_indirect_hEnv_[i,ip]*h_Env[ip,l2,l3-1]
    + sum {ip in ELEMENTS} a_indirect_mBlast[i,ip]*m_Blast[ip,l2,l3]
    + sum {j in SPECIES_Prod} a_indirect_mProd[i,j]*m_Prod[j,l2,l3]
    + a_indirect_hBlast[i]*h_Blast[l2,l3]
    + a_indirect_hOffgas[i]*h_Offgas[l2,l3]
    + a_indirect_hDCh[i]*h_DCh[l2,l3]
    + sum {ip in 0..7} a_indirect_hEnv[i,ip]*h_Env[ip,l2,l3]
    <= b_indirect[i];


#============= SECTION 4.6 : OPTIMIZATION OBJECTIVES AND GLOBAL CONSTRAINTS =============
#::::: Subsection 4.6.1 : Optimization of Nongaseous Flows and of Transition Types ::::::
#Parameters
#----------
param c_v {FLOWS_NG} default 0;
param c_BType {TYPES_PSC} default 0;
```

```
#Objective (see Subsection 5.1.4)
#---------
#maximize f:  #Chp4_Eq107_Production
#   sum{(l2,l3) in ASSIGNMENTS_CURRENT_PSC}(
#       sum{j in FLOWS_NG} c_v[j]*v[j,l2,l3]
#       + sum{k in TYPES_PSC} c_BType[k]*B_Type[k,PSCSymbol,l2,l3]);


#::::::: Subsection 4.6.2 : Limiting of Nongaseous Flows and of Transition Types ::::::::
#Sets and Related Parameters
#---------------------------
set GLOBAL_CONSTRAINTS_PSC default {};
    param a_global_v {GLOBAL_CONSTRAINTS_PSC, FLOWS_NG} default 0;
    param a_global_BType {GLOBAL_CONSTRAINTS_PSC, TYPES_PSC} default 0;
    param b_global {GLOBAL_CONSTRAINTS_PSC} default 0;

#Constraints
#-----------
subject to Chp4_Eq108_GlobalConstraints {i in GLOBAL_CONSTRAINTS_PSC}:
    sum{(l2,l3) in ASSIGNMENTS_CURRENT_PSC}(
        sum{j in FLOWS_NG} a_global_v[i,j]*v[j,l2,l3]
        + sum{k in TYPES_PSC} a_global_BType[i,k]*B_Type[k,PSCSymbol,l2,l3])
        <= b_global[i];


#==================== SECTION 5.1 : THE SINGLE-CYCLE PSC PROBLEM ========================
#::::::::: Subsection 5.1.1 : MILP Formulation of the Single-Cycle PSC Problem ::::::::::
#Parameters
#----------
param EndPreviousCycleSymbol symbolic in TYPES_PSC_Empty;
param EndCurrentCycleSymbol symbolic in TYPES_PSC_Empty diff {EndPreviousCycleSymbol};

#Constraints
#-----------
#check Chp5_Eq01 EndPreviousCycle does not have any predecessors
    check: card(TYPES_PSC_minus[EndPreviousCycleSymbol]) = 0;
#check Chp5_Eq02 EndCurrentCycle does not have any successors
    check {k in TYPES_PSC}: if (EndCurrentCycleSymbol in TYPES_PSC_minus[k])
        then 1 = 0;
subject to Chp5_Eq03_InitialFeedMassCondition {k in STREAMS_NGFeed}:
    m_Ret[k,1,0] = 0;
subject to Chp5_Eq04_InitialFeedMassCondition {j in SPECIES_NGProd}:
    m_RetProd[j,1,0] = 0;
subject to Chp5_Eq05_InitialHeatCondition:
    h_Ret[1,0] = 0;
subject to Chp5_Eq06_MeachnisticInitialCondition {k in TYPES_PSC}:
    B_Type[k,PSCSymbol,1,0] = (if k = EndPreviousCycleSymbol then 1 else 0);
subject to Chp5_Eq07_CompleteExactlyOneCycle:
    sum {k in 1..n_Asgn_max[PSCSymbol]} B_Type[EndCurrentCycleSymbol,PSCSymbol,1,k] = 1;


#:::::::::::: Subsection 5.1.2 : Critical Overlap Decomposition of a Cycle ::::::::::::::
#Sets and Related Parameters
```

```
#---------------------------
param n_PSC_system >= 1, integer;
param n_Crit_max >= 1, <= n_PSC_system - 1, integer;
set TYPES_PSC_PreCrit within TYPES_PSC diff {EndPreviousCycleSymbol};
#============================ SECTION 4.1 : GANTT STRUCTURE ============================
set TYPES_PSC_Crit within TYPES_PSC diff
        ({EndPreviousCycleSymbol} union TYPES_PSC_PreCrit);
set TYPES_PSC_PostCrit = TYPES_PSC diff
        ({EndPreviousCycleSymbol} union TYPES_PSC_PreCrit union TYPES_PSC_Crit);

#Constraints
#-----------
#check Chp5_Eq08_DisjointDecomposition is satisfied by construction
#check Chp5_Eq09 Topological Ordering
    check {k in TYPES_PSC_PreCrit}:
        card((TYPES_PSC_Crit union TYPES_PSC_PostCrit) inter TYPES_PSC_minus [k]) = 0;
#check Chp5_Eq10 Topological Ordering
    check {k in TYPES_PSC_PreCrit union TYPES_PSC_Crit}:
        card(TYPES_PSC_PostCrit inter TYPES_PSC_minus [k]) = 0;
#check Chp5_Eq11 EndCurrentCycle is either critical or postcritical
    check : if (EndCurrentCycleSymbol in (TYPES_PSC_Crit union TYPES_PSC_PostCrit))
        then 1 = 1;


#:::::::::::: Subsection 5.1.3 : Critical Overlap Decomposition of a Cycle ::::::::::::::
#Variables and Related Parameters
#--------------------------------
param d_Crit_max_actual >= 0, <= (t_max - t_Begin), default (t_max - t_Begin);
var d_Crit >=0, <= d_Crit_max_actual;
var d_Cycle >=0, <= t_max - t_Begin;
var d_CritComponent {(1,l3) in ASSIGNMENTS_CURRENT_PSC} >=0, <= d_max;
var d_CycleComponent {(1,l3) in ASSIGNMENTS_CURRENT_PSC} >=0, <= d_max;

#Constraints
#-----------
subject to Chp5_Eq12_CriticalDominance:
    d_Crit >= (n_Crit_max/n_PSC_system)*d_Cycle;
subject to Chp5_Eq13_CriticalDominance:
    d_Cycle = sum{(1,l3) in ASSIGNMENTS_CURRENT_PSC} d[PSCSymbol,1,l3];
subject to Chp5_Eq14_CriticalComponent:
    d_Crit = sum{l3 in 1..n_Asgn_max[PSCSymbol]} d_CritComponent[l3];
subject to Chp5_Eq15_CriticalComponent {(1,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_CritComponent[l3] >= d[PSCSymbol,1,l3]
                - d_max*(1 - sum{k in TYPES_PSC_Crit} B_Type[k,PSCSymbol,1,l3]);
subject to Chp5_Eq16_CriticalComponent {(1,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_CritComponent[l3] <= d_max*sum{k in TYPES_PSC_Crit} B_Type[k,PSCSymbol,1,l3];
subject to Chp5_Eq17_CriticalComponent {(1,l3) in ASSIGNMENTS_CURRENT_PSC}:
    d_CritComponent[l3] <= d[PSCSymbol,1,l3];


#:::::::::: Subsection 5.1.4 : Maximizing the Productivity of a Single Cycle ::::::::::::
#Sets and Parameters
#-------------------
param f1_star_actual >=0, default 0;
```

```
param epsilon_f1 default 0.0001*f1_star_actual;
#param d_Crit_max_actual (already implemented in previous subsection)

#Constraints
#-----------
#subject to Chp5_Eq20_dCritBound (Already implemented in previous subsection)
subject to Chp5_Eq22_MaintainProduction :
    sum{(l2,l3) in ASSIGNMENTS_CURRENT_PSC}(
        sum{j in FLOWS_NG} c_v[j]*v[j,l2,l3]
        + sum{k in TYPES_PSC} c_BType[k]*B_Type[k,PSCSymbol,l2,l3])
    >= f1_star_actual - epsilon_f1;

#Objectives
#----------
#maximize f1_divided_by_f2 (Chp5_Eq18_RatioObjective) is not supported by CPLEX.
maximize f1:  #Chp5_Eq19_Production
    sum{(l2,l3) in ASSIGNMENTS_CURRENT_PSC}(
        sum{j in FLOWS_NG} c_v[j]*v[j,l2,l3]
        + sum{k in TYPES_PSC} c_BType[k]*B_Type[k,PSCSymbol,l2,l3]);
minimize f2:  #Chp5_Eq21_CriticalDuration
    d_Crit;

#=============================================================================================
#=============================================================================================
#=============================================================================================
```

C.2 Sample *dat* Files

Section 5.2 describes the spreadsheets that automatically produce the data files that parameterize the SC-PSC problem. A total of three spreadsheets were produced, each representing different sample computations:

- A Copper PSC problem (Subsection 5.3.1), solved under the General Nickel-Copper Formulation
- A Copper PSC problem (Subsection 5.3.1), solved under the Simplified Copper Formulation
- A Nickel-Copper PSC problem (Subsection 5.3.2), solved under the General Nickel-Copper Formulation

Sample data has been prepared for each of these cases, and has been used to generate the results of Section 5.3

The following is sample data for the Copper PSC problem (Subsection 5.3.1), to parameterize the General Nickel-Copper Formulation.

```
#=============================================================================================
#=============================================================================================
#=============================================================================================
#============================ SECTION 4.1 : GANTT STRUCTURE ==================================
#:::::::::::::::::::::::::::: Subsection 4.1.1 : Assignments ::::::::::::::::::::::::::::::
set CLASSES := PSC OffgasTreatment;

param n_Asgn_max :=
    PSC                 24 #apply rule of thumb, n_Asgn_max[PSC] = ceil(2*d_max)
    OffgasTreatment      6   ;

set TYPES[PSC] := EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
                ScrapCharge EndCurrentCycle;
set TYPES[OffgasTreatment] := AssistSlagBlow AssistCopperBlow;

param PSCSymbol := PSC;
set TYPES_PSC_Empty := EndPreviousCycle EndCurrentCycle;
set TYPES_PSC_minus[EndPreviousCycle] := ;
set TYPES_PSC_minus[InitialCharge] := EndPreviousCycle;
set TYPES_PSC_minus[SlagBlow] := InitialCharge Recharge;
set TYPES_PSC_minus[Skim] := SlagBlow;
set TYPES_PSC_minus[Recharge] := Skim;
set TYPES_PSC_minus[CopperBlow] := Skim ScrapCharge;
set TYPES_PSC_minus[ScrapCharge] := Skim CopperBlow;
set TYPES_PSC_minus[EndCurrentCycle] := CopperBlow;

param t_End := 12 ;

param PreviousTypePSC := EndPreviousCycle;


#:::::::::::::::::::::::::::: Subsection 4.1.2 : Dependencies ::::::::::::::::::::::::::::
set DEPENDENCIES[PSC,SlagBlow] := (OffgasTreatment, AssistSlagBlow, 1);
set DEPENDENCIES[PSC,CopperBlow] := (OffgasTreatment, AssistCopperBlow, 1);
set DEPENDENCIES[OffgasTreatment,AssistSlagBlow] := (PSC, SlagBlow, 1);
set DEPENDENCIES[OffgasTreatment,AssistCopperBlow] := (PSC, CopperBlow, 1);


#================= SECTION 4.2 : PEIRCE-SMITH CONVERTERS AS STATE-MACHINES ==============
#:::::::::::::::::::::::: Subsections 4.2.1 : States and Transitions ::::::::::::::::::::
set STREAMS_NGFeed := FeedMatte Flux Reverts CopperScrap;
set ELEMENTS := Fe Ni Co Cu S Si Ca Al Mg O N;
set SPECIES_Prod := FeS Ni3S2 CoS Cu2S Cu_Liq Fe2SiO4 Fe3O4 NiO CoO Cu2O SiO2 CaO Al2O3
        MgO O2 N2 SO2;
set STREAMS_Prod := ConverterMatte Blister Slag Offgas;
param CMatteSymbol := ConverterMatte;
param BlisterSymbol := Blister;
param SlagSymbol := Slag;
param OffgasSymbol := Offgas;

set SPECIES[ConverterMatte] := FeS Ni3S2 CoS Cu2S;
set SPECIES[Blister] := Cu_Liq;
set SPECIES[Slag] := Fe2SiO4 Fe3O4 NiO CoO Cu2O SiO2 CaO Al2O3 MgO;
set SPECIES[Offgas] := O2 N2 SO2;
```

```
set FLOWS_NG := FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow FluxBlow
        RevertsBlow BlisterFullLadle FerroslagFullLadle CopperOxidicSlagFullLadle
        FerroslagPartialLadle BlisterPartialLadle CopperOxidicSlagPartialLadle;

set FLOWS_Ch := FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow;
set FLOWS_NGBlow := FluxBlow RevertsBlow;

set FLOWS[FeedMatte] := FeedMatteFlow;
set FLOWS[Flux] := FluxCharge FluxBlow;
set FLOWS[Reverts] := RevertsCharge RevertsBlow;
set FLOWS[CopperScrap] := CopperScrapFlow;
set FLOWS[Blister] := BlisterFullLadle BlisterPartialLadle;
set FLOWS[Slag] := FerroslagFullLadle CopperOxidicSlagFullLadle FerroslagPartialLadle
    CopperOxidicSlagPartialLadle;

set FLOWS_M_Ch := FeedMatteFlow;
set FLOWS_SM_Ch := ;
set FLOWS_M_DCh := BlisterFullLadle FerroslagFullLadle CopperOxidicSlagFullLadle;
set FLOWS_SM_DCh := BlisterPartialLadle FerroslagPartialLadle
    CopperOxidicSlagPartialLadle;

param v_PSC_max :=  1    80 ;

param rho :=        #in T/m
    FeedMatte       5.27751026355182
    Flux            2.7620882235714
    Reverts         5.52715227313848
    CopperScrap     8.92
    FeS             5.27
    Ni3S2           5.17
    CoS             5.45
    Cu2S            5.28
    Cu_Liq          7.92
    Fe2SiO4         2.5
    Fe3O4           5.2
    NiO             7.45
    CoO             5.68
    Cu2O            6
    SiO2            2.65
    CaO             3.32
    Al2O3           3.99
    MgO             3.65    ;

param m_max :=
    FeedMatte       422.200821084146
    Flux            220.967057885712
    Reverts         442.172181851078
    CopperScrap     713.6
    Fe              301.01344
    Ni              468.3368
    Co              357.376512
    Cu              713.6
    S               153.7786
```

```
    Si              99.09516
    Ca              189.821664
    Al              168.9366
    Mg              176.08768
    O               342.992240514743
    N               506.174073888917
    FeS             421.6
    Ni3S2           413.6
    CoS             436
    Cu2S            422.4
    Cu_Liq          633.6
    Fe2SiO4         200
    Fe3O4           416
    NiO             596
    CoO             454.4
    Cu2O            480
    SiO2            212
    CaO             265.6
    Al2O3           319.2
    MgO             292
    O2              9.63644202573716
    N2              506.174073888917
    SO2             366.566025644682
    ConverterMatte  436
    Blister         633.6
    Slag            596 ;

param h_min := 1    -5228457.27106541 ;
param h_max := 1    513242.078443962 ;

param u_max :=
    FeedMatteFlow                   6
    BlisterFullLadle                7
    FerroslagFullLadle              7
    CopperOxidicSlagFullLadle       7
    BlisterPartialLadle             1
    FerroslagPartialLadle           1
    CopperOxidicSlagPartialLadle    1   ;

param d_IntType_min :=
    4 SlagBlow      0.5
    4 CopperBlow    1   ;

param d_IntType_max (tr) :  0   1   2   3   4   5   6   7   :=
    EndPreviousCycle        0   0   0   0   0   0   0   0
    InitialCharge           0   0   12  0   0   0   0   0
    SlagBlow                0   0   0   0   12  0   0   0
    Skim                    0   0   0   0   0   0   12  0
    Recharge                0   0   12  0   0   0   0   0
    CopperBlow              0   0   0   0   12  0   0   0
    ScrapCharge             0   0   12  0   0   0   0   0
    EndCurrentCycle         0   0   0   0   0   0   12  0   ;

param v_Type_max (tr):  FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow
```

```
        FluxBlow RevertsBlow BlisterFullLadle FerroslagFullLadle
        CopperOxidicSlagFullLadle BlisterPartialLadle FerroslagPartialLadle
        CopperOxidicSlagPartialLadle :=
    EndPreviousCycle    0   0   0   0   0   0   0   0   0   0   0   0
    InitialCharge       80  80  80  0   0   0   0   0   0   0   0   0
    SlagBlow            0   0   0   0   80  80  0   0   0   0   0   0
    Skim                0   0   0   0   0   0   0   80  0   0   80  0
    Recharge            80  80  80  0   0   0   0   0   0   0   0   0
    CopperBlow          0   0   0   0   0   0   0   0   0   0   0   0
    ScrapCharge         0   0   0   80  0   0   0   0   0   0   0   0
    EndCurrentCycle     0   0   0   0   0   0   80  0   80  80  0   80  ;

param u_Type_max (tr): FeedMatteFlow BlisterFullLadle FerroslagFullLadle
        CopperOxidicSlagFullLadle BlisterPartialLadle FerroslagPartialLadle
        CopperOxidicSlagPartialLadle :=
    EndPreviousCycle    0   0   0   0   0   0   0
    InitialCharge       6   0   0   0   0   0   0
    SlagBlow            0   0   0   0   0   0   0
    Skim                0   0   7   0   0   1   0
    Recharge            3   0   0   0   0   0   0
    CopperBlow          0   0   0   0   0   0   0
    ScrapCharge         0   0   0   0   0   0   0
    EndCurrentCycle     0   7   0   7   1   0   1   ;

param v_u :=
    FeedMatteFlow               10
    BlisterFullLadle            10
    FerroslagFullLadle          10
    CopperOxidicSlagFullLadle   10  ;

param v_u_max :=
    BlisterPartialLadle             10
    FerroslagPartialLadle           10
    CopperOxidicSlagPartialLadle    10  ;


#:::::::::::::::::::::::: Subsection 4.2.2 : Converting Actions ::::::::::::::::::::::::
param d_Ch_Type :=
    InitialCharge   0.5
    Recharge        0.125
    ScrapCharge     0.125   ;

param d_DCh_Type :=
    Skim            0.125
    EndCurrentCycle 0.5 ;

#======================== SECTION 4.3 : INTERMEDIATE COMPUTATIONS ========================
#:::::::::::::::::::::: Subsections 4.3.1 : Intermediate Variables ::::::::::::::::::::::
param m_Blast_max :=        #in T
                    O   192.728840514743
                    N   506.174073888917 ;

param h_Blast_min :=    0   ;   #in MJ
param h_Blast_max :=    16912.618990403 ;
```

```
param h_Offgas_min :=   -1371279.42568434   ; #in MJ
param h_Offgas_max :=   656355.513247882    ;

param h_DCh_min :=  -5228457.27106541    ;  #in MJ
param h_DCh_max :=  513242.078443962 ;

param h_Env_max :=  #in MJ
    1   1296000
    2   1296000
    3   1296000
    4   1296000
    5   1296000
    6   1296000
    7   1296000 ;
#:::::::::::::::::::: Subsections 4.3.1 : Intermediate Variables ::::::::::::::::::::::
param w (tr) :  Fe Ni Co Cu S Si Ca Al Mg O N :=
    FeedMatte   0.1579 0 0 0.6 0.2421 0   0   0   0   0   0
    Flux        0   0   0   0   0   0.3973   0.03573   0.0265   0.03015    0.5103   0
    Reverts     0.2404  0   0   0.5014  0.0618  0.0599  0   0   0   0.1365  0
    CopperScrap 0   0   0   1   0   0   0   0   0   0   0
    FeS         0.6352 0   0   0   0.3648 0   0   0   0   0   0
    Ni3S2       0   0.7333 0   0   0.2670 0   0   0   0   0   0
    CoS         0   0   0.6476 0   0.3524 0   0   0   0   0   0
    Cu2S        0   0   0   0.7985 0.2015 0   0   0   0   0   0
    Cu_Liq      0   0   0   1   0   0   0   0   0   0   0
    Fe2SiO4     0.5481 0   0   0   0   0.1378 0   0   0   0.3141 0
    Fe3O4       0.7236 0   0   0   0   0   0   0   0   0.2764 0
    NiO         0   0.7858  0   0   0   0   0   0   0   0.2142  0
    CoO         0   0   0.7865 0   0   0   0   0   0   0.2135 0
    Cu2O        0   0   0   0.8882 0   0   0   0   0   0.1118 0
    SiO2        0   0   0   0   0   0.4674 0   0   0   0.5326 0
    CaO         0   0   0   0   0   0   0.7147 0   0   0.2853 0
    Al2O3       0   0   0   0   0   0   0   0.5293 0   0.4708 0
    MgO         0   0   0   0   0   0   0   0   0.6030 0.3970 0
    O2          0   0   0   0   0   0   0   0   0   1   0
    N2          0   0   0   0   0   0   0   0   0   0   1
    SO2         0   0   0   0   0.5005 0   0   0   0   0.4995 0   ;

param w_Feed_H :=       #in MJ/T
    FeedMatteFlow   200.963875139441    # evaluated at matte feed temperature
    FluxCharge      -14169.9586946575   # evaluated at cold charge temperature
    RevertsCharge   -170.182365614197   # evaluated at cold charge temperature
    CopperScrapFlow 1.9364414475923     # evaluated at cold charge temperature
    FluxBlow        -14169.9586946575   # evaluated at cold charge temperature
    RevertsBlow     -170.182365614197   ; # evaluated at cold charge temperature


#:::::::::::::::::::::: Subsection 4.3.2 : Blast Elemental Masses ::::::::::::::::::::
param m_Blast_dot :=    #in T/h
    O SlagBlow      16.0605
    N SlagBlow      42.18075
    O CopperBlow    16.0605
    N CopperBlow    42.18075 ;
```

```
#:::::::::::::::::::::: Subsection 4.3.3 : Product Species Masses ::::::::::::::::::::
set REGIMES := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime CopperBlowRegime
    CopperOverblowRegime;
set SPECIES_RgProd := FeS Ni3S2 CoS Cu2S Cu_Liq NiO CoO Cu2O;

set REGIMES_spec[FeS] := SlagBlowRegime;
set REGIMES_spec[Ni3S2] := SlagBlowRegime NickelOverblowRegime;
set REGIMES_spec[CoS] := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime;
set REGIMES_spec[Cu2S] := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime
    CopperBlowRegime;
set REGIMES_spec[Cu_Liq] := CopperBlowRegime CopperOverblowRegime;
set REGIMES_spec[NiO] := NickelOverblowRegime CobaltOverblowRegime CopperBlowRegime
    CopperOverblowRegime;
set REGIMES_spec[CoO] := CobaltOverblowRegime CopperBlowRegime CopperOverblowRegime;
set REGIMES_spec[Cu2O] := CopperOverblowRegime;

set TYPES_PSC_Prod := SlagBlow Skim Recharge CopperBlow ScrapCharge EndCurrentCycle;

param OXYGEN_EFFICIENCY :=  0.95    ;
param FERROSLAG_RATIO :=    2   ;

param OSymbol := O;
param O2Symbol := O2;
param Fe2SiO4Symbol := Fe2SiO4;
param Fe3O4Symbol := Fe3O4;


#:::::::::::::::::::::::::::: Subsection 4.3.4 : Blast Heat ::::::::::::::::::::::::::::
param h_Blast_dot :=    #in MJ/h
    SlagBlow    1409.38491586692
    CopperBlow  1409.38491586692    ;


#:::::::::::::::::::::::::::: Subsection 4.3.5 : Offgas Heat ::::::::::::::::::::::::::::
param w_Offgas_H (tr) :         O2                  N2                  SO2 :=
    EndPreviousCycle            0                   0                   0
    InitialCharge               0                   0                   0
    SlagBlow                    1240.56066795853    1296.69919323431    -3740.87975903566
    Skim                        0                   0                   0
    Recharge                    0                   0                   0
    CopperBlow                  1240.56066795853    1296.69919323431    -3740.87975903566
    ScrapCharge                 0                   0                   0
    EndCurrentCycle             0                   0                   0   ;


#:::::::::::::::::::::::::: Subsection 4.3.6 : Discharge Heat ::::::::::::::::::::::::::
param w_DCh_H : Skim:=
    FeedMatte       218.96989095771
    Flux            -12669.6798817374
    Reverts         856.638097710978
    CopperScrap     536.309107007785
    FeS             122.21539074053
```

```
    Ni3S2           961.419731551783
    CoS             927.271139828653
    Cu2S            250.984342504571
    Cu_Liq          800.160600195134
    Fe2SiO4         -5733.28422642635
    Fe3O4           -3606.67074176221
    NiO             -2378.91470463181
    CoO             -2182.36969709799
    Cu2O            -551.994700502616
    SiO2            -12640.4612884166
    CaO             -10053.5226944017
    Al2O3           -14949.489977358
    MgO             -13502.7430599066   ;


#:::::::::::::::::::: Subsection 4.3.7 : Environmental Heat Losses ::::::::::::::::::::
param h_EnvType_dot (tr):                                                       #in MJ/h
                        0       1       2       3       4       5       6       7   :=
    EndPreviousCycle    0       0       0       0       0       0       0       0
    InitialCharge       0       0       0       0       0       0       0       0
    SlagBlow            108000  108000  108000  108000  108000  108000  108000  108000
    Skim                108000  108000  108000  108000  108000  108000  108000  108000
    Recharge            108000  108000  108000  108000  108000  108000  108000  108000
    CopperBlow          27000   27000   27000   27000   27000   27000   27000   27000
    ScrapCharge         27000   27000   27000   27000   27000   27000   27000   27000
    EndCurrentCycle     0       0       0       0       0       0       0       0       ;


#========================== SECTION 4.4 : FORWARD COMPUTATIONS ==========================
#:::::::::::::::::::: Subsection 4.4.1 : Retained Elemental Masses ::::::::::::::::::::
#:::::::::::::::::::::: Subsection 4.4.2 : Retained Feed Masses ::::::::::::::::::::::
#:::::::::::::::::::: Subsection 4.4.3 : Forward Heat Computation ::::::::::::::::::::

#====================== SECTION 4.5 : FEASIBLE CONVERTER TRANSITIONS ====================
#:::::::: Subsection 4.5.1 : Direct Transition Constraints in General Linear Form :::::::

#:::::::::::::::::::: Subsection 4.5.2 : Bath Composition Constraints ::::::::::::::::::
param w_Bath_Blow_max :=
    #SlagBlow implies no Blister
        Blister SlagBlow 0;

param w_Bath_O_max  :=
    #CopperBlow implies no initial Fe
        Fe CopperBlow 0

    #ChargeScrap implies no initial Fe
        Fe ScrapCharge 0

    #EndCurrentCycle implies no initial S
        S EndCurrentCycle 0;

param w_strm_O_max :=
    #Restricted use of excess flux
        SiO2 Slag Skim 0.1 ;
```

```
#:::::::::::::::::::::: Subsection 4.5.3 : Volume Constraints ::::::::::::::::::::::

#:::::::::::::::::::: Subsection 4.5.4 : Temperature Constraints ::::::::::::::::::::
param w_Blow_H_min : EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
        ScrapCharge EndCurrentCycle := #in MJ/T
    FeedMatte       -501.27 -501.27 110.93 -501.27 -501.27 170.95 -501.27 -501.27
    Flux            -14170 -14170 -12951 -14170 -14170 -12797 -14170 -14170
    Reverts         -170.18 -170.18 659.15 -170.18 -170.18 767.05 -170.18 -170.18
    CopperScrap     1.9364 1.9364 447.31 1.9364 1.9364 496.37 1.9364 1.9364
    FeS             -731.63 -731.63 -5.8608 -731.63 -731.63 65.293 -731.63 -731.63
    Ni3S2           3.2839  3.2839  817.70  3.2839  3.2839  897.54  3.2839 3.2839
    CoS             4.1759 4.1759 788.81 4.1759 4.1759 865.73 4.1759 4.1759
    Cu2S            -425.05 -425.05 149.58 -425.05 -425.05 205.92 -425.05 -425.05
    Cu_Liq          207.33 207.33 711.24 207.33 207.33 760.64 207.33 207.33
    Fe2SiO4         -7150.0 -7150.0 -5945.8 -7150.0 -7150.0 -5827.7 -7150.0 -7150.0
    Fe3O4           -4821.9 -4821.9 -3825.6 -4821.9 -4821.9 -3705.4 -4821.9 -4821.9
    NiO             -3269.4 -3269.4 -2523.5 -3269.4 -3269.4 -2443.7 -3269.4 -3269.4
    CoO             -3206.4 -3206.4 -2339.1 -3206.4 -3206.4 -2252.7 -3206.4 -3206.4
    Cu2O            -1255.4 -1255.4 -672.81 -1255.4 -1255.4 -606.36 -1255.4 -1255.4
    SiO2            -14163 -14163 -12930 -14163 -14163 -12771 -14163 -14163
    CaO             -11320 -11320 -10279 -11320 -11320 -10155 -11320 -11320
    Al2O3           -16380 -16380 -15204 -16380 -16380 -15064 -16380 -16380
    MgO             -14934 -14934 -13735 -14934 -14934 -13607 -14934 -14934     ;

param w_Blow_H_max : EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
        ScrapCharge EndCurrentCycle := #in MJ/T
    FeedMatte   230.97 230.97 230.97 230.97 230.97 230.97 230.97 230.97
    Flux        -12637 -12637 -12637 -12637 -12637 -12637 -12637 -12637
    Reverts     879.45 879.49 879.49 879.49 879.49 879.49 879.49 879.49
    CopperScrap 546.39 546.39 546.39 546.39 546.39 546.39 546.39 546.39
    FeS         136.45 136.45 136.45 136.45 136.45 136.45 136.45 136.45
    Ni3S2       977.39 977.39 977.39 977.39 977.39 977.39 977.39 977.39
    CoS         942.66 942.66 942.66 942.66 942.66 942.66 942.66 942.66
    Cu2S        262.25 262.25 262.25 262.25 262.25 262.25 262.25 262.25
    Cu_Liq      810.04 810.04 810.04 810.04 810.04 810.04 810.04 810.04
    Fe2SiO4     -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7
    Fe3O4       -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6
    NiO         -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6
    CoO         -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6
    Cu2O        -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24
    SiO2        -12607 -12607 -12607 -12607 -12607 -12607 -12607 -12607
    CaO         -10028 -10028 -10028 -10028 -10028 -10028 -10028 -10028
    Al2O3       -14920 -14920 -14920 -14920 -14920 -14920 -14920 -14920
    MgO         -13477 -13477 -13477 -13477 -13477 -13477 -13477 -13477 ;

param w_DCh_H_min :     EndCurrentCycle := #in MJ/T
    FeedMatte           170.953848775659
    Flux                -12797.0498513263
    Reverts             767.053917844114
    CopperScrap         496.36990997793
    FeS                 65.2926288249346
    Ni3S2               897.544009371792
```

```
    CoS                 865.731457640451
    Cu2S                205.915385437021
    Cu_Liq              760.638364334498
    Fe2SiO4             -5827.73437710866
    Fe3O4               -3705.37822518764
    NiO                 -2443.67823153721
    CoO                 -2252.71768244752
    Cu2O                -606.357787312864
    SiO2                -12771.4558084642
    CaO                 -10155.2236001187
    Al2O3               -15064.3120428775
    MgO                 -13606.7126396395   ;

param w_DCh_H_max :     EndCurrentCycle :=  #in MJ/T
    FeedMatte           230.973901503222
    Flux                -12637.2684712361
    Reverts             879.486290568683
    CopperScrap         546.390208779963
    FeS                 136.446081219429
    Ni3S2               977.38866209678
    CoS                 942.656060375703
    Cu2S                262.251581771458
    Cu_Liq              810.041159160293
    Fe2SiO4             -5709.67168875578
    Fe3O4               -3581.64276106801
    NiO                 -2362.60333645964
    CoO                 -2164.59758386118
    Cu2O                -538.237252116479
    SiO2                -12607.0954189282
    CaO                 -10027.7312463732
    Al2O3               -14920.403009648
    MgO                 -13476.6130469213   ;


#::::::: Subsection 4.5.5 : Indirect Transition Constraints in General Linear Form ::::::
set INDIRECT_TRANSITION_CONSTRAINTS_PSC := CopperBlowMaximumOxidize
    EndCurrentCycleMinimumOxidize;

param a_indirect_mRetProd_ :=
    #EndCurrentCycleMinimumOxidize
        EndCurrentCycleMinimumOxidize   Cu_Liq      0.05
        EndCurrentCycleMinimumOxidize   Cu2O        -0.8437805  ;

param a_indirect_BType :=
    #CopperBlowMaximumOxidize
        CopperBlowMaximumOxidize        CopperBlow      405.01464
    #EndCurrentCycleMinimumOxidize
        EndCurrentCycleMinimumOxidize   EndCurrentCycle 31.68       ;

param a_indirect_mProd :=
    #CopperBlowMaximumOxidize
        CopperBlowMaximumOxidize    Cu_Liq  -0.05
        CopperBlowMaximumOxidize    Cu2O    0.8437805   ;
```

```
param b_indirect :=
    CopperBlowMaximumOxidize        405.01464
    EndCurrentCycleMinimumOxidize   31.68       ;


#============= SECTION 4.6 : OPTIMIZATION OBJECTIVES AND GLOBAL CONSTRAINTS =============
#:::::: Subsection 4.6.1 : Optimization of Nongaseous Flows and of Transition Types :::::
param c_v :=    FeedMatteFlow   1;

#:::::::: Subsection 4.6.2 : Limiting of Nongaseous Flows and of Transition Types :::::::
set GLOBAL_CONSTRAINTS_PSC := MaximumRecharges MaximumScrapCharges;

param a_global_BType :=
    #Limit the number of recharges
        MaximumRecharges    Recharge    1
    #Limit the number of scrap charges
        MaximumScrapCharges ScrapCharge 1;

param b_global :=
    #Limit the number of recharges
        MaximumRecharges    2
    #Limit the number of scrap charges
        MaximumScrapCharges 2   ;


#==================== SECTION 5.1 : THE SINGLE-CYCLE PSC PROBLEM ========================
#:::::::: Subsection 5.1.1 : MILP Formulation of the Single-Cycle PSC Problem ::::::::::
param EndPreviousCycleSymbol := EndPreviousCycle;
param EndCurrentCycleSymbol := EndCurrentCycle;


#:::::::::: Subsection 5.1.2 : Critical Overlap Decomposition of a Cycle ::::::::::::::
param n_PSC_system := 2 ;
param n_Crit_max := 1 ;

set TYPES_PSC_PreCrit := InitialCharge;
set TYPES_PSC_Crit := SlagBlow Skim Recharge CopperBlow ScrapCharge;


#:::::::::::: Subsection 5.1.3 : Critical Overlap Decomposition of a Cycle ::::::::::::

#:::::::::: Subsection 5.1.4 : Maximizing the Productivity of a Single Cycle ::::::::::::

#=============================================================================================
#=============================================================================================
#=============================================================================================
```

The following is sample data for the Copper PSC problem (Subsection 5.3.1), to parameterize the Simplified Copper PSC Formulation.

```
#=============================================================================================
```

```
#=============================================================================================
#=============================================================================================
#============================ SECTION 4.1 : GANTT STRUCTURE ==================================
#:::::::::::::::::::::::::::: Subsection 4.1.1 : Assignments ::::::::::::::::::::::::::::::
set CLASSES := PSC OffgasTreatment;

param n_Asgn_max :=
    PSC             24  #apply rule of thumb, n_Asgn_max[PSC] = ceil(2*d_max)
    OffgasTreatment 6   ;

set TYPES[PSC] := EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
    ScrapCharge EndCurrentCycle;
set TYPES[OffgasTreatment] := AssistSlagBlow AssistCopperBlow;

param PSCSymbol := PSC;
set TYPES_PSC_Empty := EndPreviousCycle EndCurrentCycle;
set TYPES_PSC_minus[EndPreviousCycle] := ;
set TYPES_PSC_minus[InitialCharge] := EndPreviousCycle;
set TYPES_PSC_minus[SlagBlow] := InitialCharge Recharge;
set TYPES_PSC_minus[Skim] := SlagBlow;
set TYPES_PSC_minus[Recharge] := Skim;
set TYPES_PSC_minus[CopperBlow] := Skim ScrapCharge;
set TYPES_PSC_minus[ScrapCharge] := Skim CopperBlow;
set TYPES_PSC_minus[EndCurrentCycle] := CopperBlow;

param t_End := 12 ;

param PreviousTypePSC := EndPreviousCycle;


#:::::::::::::::::::::::::::: Subsection 4.1.2 : Dependencies ::::::::::::::::::::::::::::
set DEPENDENCIES[PSC,SlagBlow] := (OffgasTreatment, AssistSlagBlow, 1);
set DEPENDENCIES[PSC,CopperBlow] := (OffgasTreatment, AssistCopperBlow, 1);
set DEPENDENCIES[OffgasTreatment,AssistSlagBlow] := (PSC, SlagBlow, 1);
set DEPENDENCIES[OffgasTreatment,AssistCopperBlow] := (PSC, CopperBlow, 1);


#=============== SECTION 4.2 : PEIRCE-SMITH CONVERTERS AS STATE-MACHINES ================
#:::::::::::::::::::::: Subsections 4.2.1 : States and Transitions ::::::::::::::::::::::
set STREAMS_NGFeed := FeedMatte Flux Reverts CopperScrap;
set ELEMENTS := Fe Cu S Si Ca Al Mg O N;
set SPECIES_Prod := FeS Cu2S Cu_Liq Fe2SiO4 Fe3O4 Cu2O SiO2 CaO Al2O3 MgO O2 N2 SO2;
set STREAMS_Prod := ConverterMatte Blister Slag Offgas;
param CMatteSymbol := ConverterMatte;
param BlisterSymbol := Blister;
param SlagSymbol := Slag;
param OffgasSymbol := Offgas;

set SPECIES[ConverterMatte] := FeS Cu2S;
set SPECIES[Blister] := Cu_Liq;
set SPECIES[Slag] := Fe2SiO4 Fe3O4 Cu2O SiO2 CaO Al2O3 MgO;
set SPECIES[Offgas] := O2 N2 SO2;

set FLOWS_NG := FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow FluxBlow
```

```
    RevertsBlow BlisterFullLadle FerroslagFullLadle CopperOxidicSlagFullLadle
    FerroslagPartialLadle BlisterPartialLadle CopperOxidicSlagPartialLadle;

set FLOWS_Ch := FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow;
set FLOWS_NGBlow := FluxBlow RevertsBlow;

set FLOWS[FeedMatte] := FeedMatteFlow;
set FLOWS[Flux] := FluxCharge FluxBlow;
set FLOWS[Reverts] := RevertsCharge RevertsBlow;
set FLOWS[CopperScrap] := CopperScrapFlow;
set FLOWS[Blister] := BlisterFullLadle BlisterPartialLadle;
set FLOWS[Slag] := FerroslagFullLadle CopperOxidicSlagFullLadle FerroslagPartialLadle
    CopperOxidicSlagPartialLadle;

set FLOWS_M_Ch := FeedMatteFlow;
set FLOWS_SM_Ch := ;
set FLOWS_M_DCh := BlisterFullLadle FerroslagFullLadle CopperOxidicSlagFullLadle;
set FLOWS_SM_DCh := BlisterPartialLadle FerroslagPartialLadle
    CopperOxidicSlagPartialLadle;

param v_PSC_max := 1    80  ;

param rho :=        #in T/m
    FeedMatte       5.27751026355182
    Flux            2.7620882235714
    Reverts         5.52715227313848
    CopperScrap     8.92
    FeS             5.27
    Cu2S            5.28
    Cu_Liq          7.92
    Fe2SiO4         2.5
    Fe3O4           5.2
    Cu2O            6
    SiO2            2.65
    CaO             3.32
    Al2O3           3.99
    MgO             3.65;

param m_max :=
    FeedMatte       422.200821084146
    Flux            220.967057885712
    Reverts         442.172181851078
    CopperScrap     713.6
    Fe              301.01344
    Cu              713.6
    S               153.7786
    Si              99.09516
    Ca              189.821664
    Al              168.9366
    Mg              176.08768
    O               342.992240514743
    N               506.174073888917
    FeS             421.6
    Cu2S            422.4
```

```
    Cu_Liq          633.6
    Fe2SiO4         200
    Fe3O4           416
    Cu2O            480
    SiO2            212
    CaO             265.6
    Al2O3           319.2
    MgO             292
    O2              9.63644202573716
    N2              506.174073888917
    SO2             366.566025644682
    ConverterMatte  422.4
    Blister         633.6
    Slag            480 ;

param h_min := 1    -5228457.27106541   ;
param h_max := 1    513242.078443962    ;

param u_max :=
    FeedMatteFlow                   6
    BlisterFullLadle                7
    FerroslagFullLadle              7
    CopperOxidicSlagFullLadle       7
    BlisterPartialLadle             1
    FerroslagPartialLadle           1
    CopperOxidicSlagPartialLadle    1   ;

param d_IntType_min :=
    4 SlagBlow      0.5
    4 CopperBlow    1   ;

param d_IntType_max (tr) :  0   1   2   3   4   5   6   7   :=
    EndPreviousCycle        0   0   0   0   0   0   0   0
    InitialCharge           0   0   12  0   0   0   0   0
    SlagBlow                0   0   0   0   12  0   0   0
    Skim                    0   0   0   0   0   0   12  0
    Recharge                0   0   12  0   0   0   0   0
    CopperBlow              0   0   0   0   12  0   0   0
    ScrapCharge             0   0   12  0   0   0   0   0
    EndCurrentCycle         0   0   0   0   0   0   12  0   ;

param v_Type_max (tr): FeedMatteFlow FluxCharge RevertsCharge CopperScrapFlow FluxBlow
        RevertsBlow BlisterFullLadle FerroslagFullLadle CopperOxidicSlagFullLadle
        BlisterPartialLadle FerroslagPartialLadle CopperOxidicSlagPartialLadle :=
    EndPreviousCycle    0   0   0   0   0   0   0   0   0   0   0   0
    InitialCharge       80  80  80  0   0   0   0   0   0   0   0   0
    SlagBlow            0   0   0   0   80  80  0   0   0   0   0   0
    Skim                0   0   0   0   0   0   0   80  0   0   80  0
    Recharge            80  80  80  0   0   0   0   0   0   0   0   0
    CopperBlow          0   0   0   0   0   0   0   0   0   0   0   0
    ScrapCharge         0   0   0   80  0   0   0   0   0   0   0   0
    EndCurrentCycle     0   0   0   0   0   0   80  0   80  80  0   80  ;

param u_Type_max (tr): FeedMatteFlow BlisterFullLadle FerroslagFullLadle
```

```
        CopperOxidicSlagFullLadle BlisterPartialLadle FerroslagPartialLadle
        CopperOxidicSlagPartialLadle    :=
    EndPreviousCycle    0   0   0   0   0   0   0
    InitialCharge       6   0   0   0   0   0   0
    SlagBlow            0   0   0   0   0   0   0
    Skim                0   0   7   0   0   1   0
    Recharge            3   0   0   0   0   0   0
    CopperBlow          0   0   0   0   0   0   0
    ScrapCharge         0   0   0   0   0   0   0
    EndCurrentCycle     0   7   0   7   1   0   1   ;

param v_u :=
    FeedMatteFlow               10
    BlisterFullLadle            10
    FerroslagFullLadle          10
    CopperOxidicSlagFullLadle   10  ;

param v_u_max :=
    BlisterPartialLadle             10
    FerroslagPartialLadle           10
    CopperOxidicSlagPartialLadle    10  ;


#:::::::::::::::::::::::: Subsection 4.2.2 : Converting Actions ::::::::::::::::::::::::
param d_Ch_Type :=
    InitialCharge       0.5
    Recharge            0.125
    ScrapCharge         0.125   ;

param d_DCh_Type :=
    Skim                0.125
    EndCurrentCycle     0.5 ;


#======================= SECTION 4.3 : INTERMEDIATE COMPUTATIONS =======================
#::::::::::::::::::::: Subsections 4.3.1 : Intermediate Variables :::::::::::::::::::::
param m_Blast_max :=    #in T
    O   192.728840514743
    N   506.174073888917    ;

param h_Blast_min :=    0   ;   #in MJ
param h_Blast_max :=    16912.618990403 ;

param h_Offgas_min :=   -1371279.42568434   ; #in MJ
param h_Offgas_max :=   656355.513247882    ;

param h_DCh_min :=  -5228457.27106541   ; #in MJ
param h_DCh_max :=  513242.078443962    ;

param h_Env_max :=  #in MJ
    1   1296000
    2   1296000
    3   1296000
    4   1296000
```

```
    5   1296000
    6   1296000
    7   1296000 ;

param w (tr) : Fe Cu S Si Ca Al Mg O N :=
    FeedMatte       0.1579 0.6 0.2421 0 0 0 0 0 0
    Flux            0 0 0 0.3973 0.03573 0.0265 0.03015 0.5103 0
    Reverts         0.2404 0.5014 0.0618 0.0599 0 0 0 0.1365 0
    CopperScrap     0 1 0 0 0 0 0 0 0
    FeS             0.6352 0 0.3648 0 0 0 0 0 0
    Cu2S            0 0.7985 0.2015 0 0 0 0 0 0
    Cu_Liq          0 1 0 0 0 0 0 0 0
    Fe2SiO4         0.5481 0 0 0.1378 0 0 0 0.3141 0
    Fe3O4           0.7236 0 0 0 0 0 0 0.2764 0
    Cu2O            0 0.8882 0 0 0 0 0 0.1118 0
    SiO2            0 0 0 0.4674 0 0 0 0.5326 0
    CaO             0 0 0 0 0.7147 0 0 0.2853 0
    Al2O3           0 0 0 0 0 0.5293 0 0.4707 0
    MgO             0 0 0 0 0 0 0.6030 0.3970 0
    O2              0 0 0 0 0 0 0 1 0
    N2              0 0 0 0 0 0 0 0 1
    SO2             0 0 0.5005 0 0 0 0 0.4995 0 ;

param w_Feed_H :=       #in MJ/T
    FeedMatteFlow       200.963875139441    # evaluated at matte feed temperature
    FluxCharge          -14169.9586946575   # evaluated at cold charge temperature
    RevertsCharge       -170.182365614197   # evaluated at cold charge temperature
    CopperScrapFlow     1.9364414475923     # evaluated at cold charge temperature
    FluxBlow            -14169.9586946575   # evaluated at cold charge temperature
    RevertsBlow         -170.182365614197   ; # evaluated at cold charge temperature


#:::::::::::::::::::::: Subsection 4.3.2 : Blast Elemental Masses ::::::::::::::::::::
param m_Blast_dot :=    #in T/h
    O SlagBlow          16.0605
    N SlagBlow          42.18075
    O CopperBlow        16.0605
    N CopperBlow        42.18075    ;


#:::::::::::::::::::::: Subsection 4.3.3 : Product Species Masses ::::::::::::::::::::
set REGIMES := SlagBlowRegime CopperBlowRegime CopperOverblowRegime;
set SPECIES_RgProd := FeS Cu2S Cu_Liq Cu2O;

set REGIMES_spec[FeS] := SlagBlowRegime;
set REGIMES_spec[Cu2S] := SlagBlowRegime CopperBlowRegime;
set REGIMES_spec[Cu_Liq] := CopperBlowRegime CopperOverblowRegime;
set REGIMES_spec[Cu2O] := CopperOverblowRegime;

set TYPES_PSC_Prod :=  SlagBlow Skim Recharge CopperBlow ScrapCharge EndCurrentCycle;

param OXYGEN_EFFICIENCY := 0.95 ;
param FERROSLAG_RATIO := 2 ;
```

```
param OSymbol := O;
param O2Symbol := O2;
param Fe2SiO4Symbol := Fe2SiO4;
param Fe3O4Symbol := Fe3O4;


#:::::::::::::::::::::::::: Subsection 4.3.4 : Blast Heat ::::::::::::::::::::::::::
param h_Blast_dot := #in MJ/h
    SlagBlow    1409.38491586692
    CopperBlow  1409.38491586692    ;


#:::::::::::::::::::::::::: Subsection 4.3.5 : Offgas Heat ::::::::::::::::::::::::::
param w_Offgas_H (tr) : O2                  N2                  SO2                 :=
    EndPreviousCycle    0                   0                   0
    InitialCharge       0                   0                   0
    SlagBlow            1240.56066795853    1296.69919323431    -3740.87975903566
    Skim                0                   0                   0
    Recharge            0                   0                   0
    CopperBlow          1240.56066795853    1296.69919323431    -3740.87975903566
    ScrapCharge         0                   0                   0
    EndCurrentCycle     0                   0                   0   ;


#:::::::::::::::::::::::::: Subsection 4.3.6 : Discharge Heat ::::::::::::::::::::::::::
param w_DCh_H : Skim :=
    FeedMatte   218.96989095771
    Flux        -12669.6798817374
    Reverts     856.638097710978
    CopperScrap 536.309107007785
    FeS         122.21539074053
    Cu2S        250.984342504571
    Cu_Liq      800.160600195134
    Fe2SiO4     -5733.28422642635
    Fe3O4       -3606.67074176221
    Cu2O        -551.994700502616
    SiO2        -12640.4612884166
    CaO         -10053.5226944017
    Al2O3       -14949.489977358
    MgO         -13502.7430599066 ;


#:::::::::::::::::::: Subsection 4.3.7 : Environmental Heat Losses ::::::::::::::::::::
param h_EnvType_dot (tr):                                                       #in MJ/h
                        0       1       2       3       4       5       6       7   :=
    EndPreviousCycle    0       0       0       0       0       0       0       0
    InitialCharge       0       0       0       0       0       0       0       0
    SlagBlow            108000  108000  108000  108000  108000  108000  108000  108000
    Skim                108000  108000  108000  108000  108000  108000  108000  108000
    Recharge            108000  108000  108000  108000  108000  108000  108000  108000
    CopperBlow          27000   27000   27000   27000   27000   27000   27000   27000
    ScrapCharge         27000   27000   27000   27000   27000   27000   27000   27000
    EndCurrentCycle     0       0       0       0       0       0       0       0       ;
```

```
#========================= SECTION 4.4 : FORWARD COMPUTATIONS ==========================
#:::::::::::::::::::: Subsection 4.4.1 : Retained Elemental Masses ::::::::::::::::::::
#:::::::::::::::::::::: Subsection 4.4.2 : Retained Feed Masses ::::::::::::::::::::::
#:::::::::::::::::::::: Subsection 4.4.3 : Forward Heat Computation ::::::::::::::::::::

#=================== SECTION 4.5 : FEASIBLE CONVERTER TRANSITIONS ======================
#:::::::: Subsection 4.5.1 : Direct Transition Constraints in General Linear Form :::::::

#:::::::::::::::::::: Subsection 4.5.2 : Bath Composition Constraints ::::::::::::::::::
param w_Bath_Blow_max :=
    #SlagBlow implies no Blister
        Blister SlagBlow 0;

param w_Bath_O_max  :=
    #CopperBlow implies no initial Fe
        Fe CopperBlow 0

    #ChargeScrap implies no initial Fe
        Fe ScrapCharge 0

    #EndCurrentCycle implies no initial S
        S EndCurrentCycle 0;

param w_strm_O_max :=
    #Restricted use of excess flux
        SiO2 Slag Skim 0.1 ;


#:::::::::::::::::::::: Subsection 4.5.3 : Volume Constraints ::::::::::::::::::::::::

#:::::::::::::::::::: Subsection 4.5.4 : Temperature Constraints ::::::::::::::::::::::
param w_Blow_H_min : EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
        ScrapCharge EndCurrentCycle := #in MJ/T
    FeedMatte       -501.27 -501.27 110.93 -501.27 -501.27 170.95 -501.27 -501.27
    Flux            -14170 -14170 -12951 -14170 -14170 -12797 -14170 -14170
    Reverts         -170.18 -170.18 659.15 -170.18 -170.18 767.05 -170.18 -170.18
    CopperScrap     1.9364 1.9364 447.31 1.9364 1.9364 496.37 1.9364 1.9364
    FeS             -731.63 -731.63 -5.8608 -731.63 -731.63 65.293 -731.63 -731.63
    Cu2S            -425.05 -425.05 149.58 -425.05 -425.05 205.92 -425.05 -425.05
    Cu_Liq          207.33 207.33 711.24 207.33 207.33 760.64 207.33 207.33
    Fe2SiO4         -7150.0 -7150.0 -5945.8 -7150.0 -7150.0 -5827.7 -7150.0 -7150.0
    Fe3O4           -4821.9 -4821.9 -3825.6 -4821.9 -4821.9 -3705.4 -4821.9 -4821.9
    Cu2O            -1255.4 -1255.4 -672.81 -1255.4 -1255.4 -606.36 -1255.4 -1255.4
    SiO2            -14163 -14163 -12930 -14163 -14163 -12771 -14163 -14163
    CaO             -11320 -11320 -10279 -11320 -11320 -10155 -11320 -11320
    Al2O3           -16370 -16380 -15204 -16380 -16380 -15064 -16380 -16380
    MgO             -14934 -14934 -13735 -14934 -14934 -13607 -14934 -14934 ;

param w_Blow_H_max : EndPreviousCycle InitialCharge SlagBlow Skim Recharge CopperBlow
        ScrapCharge EndCurrentCycle := #in MJ/T
    FeedMatte       230.97 230.97 230.97 230.97 230.97 230.97 230.97 230.97
    Flux            -12637 -12637 -12637 -12637 -12637 -12637 -12637 -12637
    Reverts         879.49 879.49 879.49 879.49 879.49 879.49 879.49 879.49
    CopperScrap     546.39 546.39 546.39 546.39 546.39 546.39 546.39 546.39
```

```
    FeS             136.45 136.45 136.45 136.45 136.45 136.45 136.45 136.45
    Cu2S            262.25 262.25 262.25 262.25 262.25 262.25 262.25 262.25
    Cu_Liq          810.04 810.04 810.04 810.04 810.04 810.04 810.04 810.04
    Fe2SiO4         -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7
    Fe3O4           -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6
    Cu2O            -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24
    SiO2            -12607 -12607 -12607 -12607 -12607 -12607 -12607 -12607
    CaO             -10028 -10028 -10028 -10028 -10028 -10028 -10028 -10028
    Al2O3           -14920 -14920 -14920 -14920 -14920 -14920 -14920 -14920
    MgO             -13477 -13477 -13477 -13477 -13477 -13477 -13477 -13477 ;

param w_DCh_H_min : EndCurrentCycle :=  #in MJ/T
    FeedMatte       170.953848775659
    Flux            -12797.0498513263
    Reverts         767.053917844114
    CopperScrap     496.36990997793
    FeS             65.2926288249346
    Cu2S            205.915385437021
    Cu_Liq          760.638364334498
    Fe2SiO4         -5827.73437710866
    Fe3O4           -3705.37822518764
    Cu2O            -606.357787312864
    SiO2            -12771.4558084642
    CaO             -10155.2236001187
    Al2O3           -15064.3120428775
    MgO             -13606.7126396395   ;

param w_DCh_H_max : EndCurrentCycle :=  #in MJ/T
    FeedMatte       230.973901503222
    Flux            -12637.2684712361
    Reverts         879.486290568683
    CopperScrap     546.390208779963
    FeS             136.446081219429
    Cu2S            262.251581771458
    Cu_Liq          810.041159160293
    Fe2SiO4         -5709.67168875578
    Fe3O4           -3581.64276106801
    Cu2O            -538.237252116479
    SiO2            -12607.0954189282
    CaO             -10027.7312463732
    Al2O3           -14920.403009648
    MgO             -13476.6130469213   ;


#::::::: Subsection 4.5.5 : Indirect Transition Constraints in General Linear Form ::::::
set INDIRECT_TRANSITION_CONSTRAINTS_PSC := CopperBlowMaximumOxidize
     EndCurrentCycleMinimumOxidize;

param a_indirect_mRetProd_ :=
    #EndCurrentCycleMinimumOxidize
        EndCurrentCycleMinimumOxidize   Cu_Liq  0.05
        EndCurrentCycleMinimumOxidize   Cu2O    -0.8437805  ;

param a_indirect_BType :=
```

```
    #CopperBlowMaximumOxidize
        CopperBlowMaximumOxidize    CopperBlow  405.01464
    #EndCurrentCycleMinimumOxidize
        EndCurrentCycleMinimumOxidize   EndCurrentCycle 31.68   ;

param a_indirect_mProd :=
    #CopperBlowMaximumOxidize
        CopperBlowMaximumOxidize    Cu_Liq  -0.05
        CopperBlowMaximumOxidize    Cu2O    0.8437805   ;

param b_indirect :=
    CopperBlowMaximumOxidize        405.01464
    EndCurrentCycleMinimumOxidize   31.68       ;


#=========== SECTION 4.6 : OPTIMIZATION OBJECTIVES AND GLOBAL CONSTRAINTS ==============
#::::: Subsection 4.6.1 : Optimization of Nongaseous Flows and of Transition Types ::::::
param c_v :=    FeedMatteFlow   1;

#::::::: Subsection 4.6.2 : Limiting of Nongaseous Flows and of Transition Types ::::::::
set GLOBAL_CONSTRAINTS_PSC := MaximumRecharges MaximumScrapCharges;

param a_global_BType :=
    #Limit the number of recharges
        MaximumRecharges Recharge 1
    #Limit the number of scrap charges
        MaximumScrapCharges ScrapCharge 1;

param b_global :=
    #Limit the number of recharges
        MaximumRecharges    2
    #Limit the number of scrap charges
        MaximumScrapCharges 2   ;


#==================== SECTION 5.1 : THE SINGLE-CYCLE PSC PROBLEM ========================
#::::::::: Subsection 5.1.1 : MILP Formulation of the Single-Cycle PSC Problem :::::::::
param EndPreviousCycleSymbol := EndPreviousCycle;
param EndCurrentCycleSymbol := EndCurrentCycle;


#:::::::::::: Subsection 5.1.2 : Critical Overlap Decomposition of a Cycle ::::::::::::::
param n_PSC_system := 2 ;
param n_Crit_max := 1 ;

set TYPES_PSC_PreCrit := InitialCharge;
set TYPES_PSC_Crit := SlagBlow Skim Recharge CopperBlow ScrapCharge;


#:::::::::::: Subsection 5.1.3 : Critical Overlap Decomposition of a Cycle ::::::::::::::

#:::::::: Subsection 5.1.4 : Maximizing the Productivity of a Single Cycle ::::::::::::::

#=========================================================================================
```

```
#=============================================================================================
#=============================================================================================
```

The following is sample data for the Nickel-Copper PSC problem (Subsection 5.3.2), to parameterize the General Nickel-Copper PSC Formulation.

```
#=============================================================================================
#=============================================================================================
#=============================================================================================
#============================ SECTION 4.1 : GANTT STRUCTURE ==============================
#:::::::::::::::::::::::::: Subsection 4.1.1 : Assignments ::::::::::::::::::::::::::::::
set CLASSES := PSC SmeltingFurnace;

param n_Asgn_max :=
        PSC             24  #apply rule of thumb, n_Asgn_max[PSC] = ceil(2*d_max)
        SmeltingFurnace 3   ;

set TYPES[PSC] := EndPreviousCycle  InitialCharge  SlagBlow  Skim  Recharge
    SlagBlowAndSkimWithoutAnyMoreFeedMatte ExtendProductionCycleWithoutAnyMoreFeedMatte
    EndCurrentCycle;
set TYPES[SmeltingFurnace] := AssistPSCInitialCharge AssistPSCRecharge;

param PSCSymbol := PSC;
set TYPES_PSC_Empty := EndPreviousCycle EndCurrentCycle;
set TYPES_PSC_minus[EndPreviousCycle] := ;
set TYPES_PSC_minus[InitialCharge] := EndPreviousCycle;
set TYPES_PSC_minus[SlagBlow] := InitialCharge, Recharge;
set TYPES_PSC_minus[Skim] := SlagBlow;
set TYPES_PSC_minus[Recharge] := Skim;
set TYPES_PSC_minus[SlagBlowAndSkimWithoutAnyMoreFeedMatte] := InitialCharge;
set TYPES_PSC_minus[ExtendProductionCycleWithoutAnyMoreFeedMatte] := Skim
    SlagBlowAndSkimWithoutAnyMoreFeedMatte ExtendProductionCycleWithoutAnyMoreFeedMatte;
set TYPES_PSC_minus[EndCurrentCycle] := Skim SlagBlowAndSkimWithoutAnyMoreFeedMatte
    ExtendProductionCycleWithoutAnyMoreFeedMatte;

param t_End := 12 ;

param PreviousTypePSC := EndPreviousCycle;


#:::::::::::::::::::::::::: Subsection 4.1.2 : Dependencies ::::::::::::::::::::::::::::
set DEPENDENCIES[PSC,InitialCharge] := (SmeltingFurnace, AssistPSCInitialCharge, 1);
set DEPENDENCIES[PSC,Recharge] := (SmeltingFurnace, AssistPSCRecharge, 1);
set DEPENDENCIES[SmeltingFurnace,AssistPSCInitialCharge] := (PSC, InitialCharge, 1);
set DEPENDENCIES[SmeltingFurnace,AssistPSCRecharge] := (PSC, Recharge, 1);


#=============== SECTION 4.2 : PEIRCE-SMITH CONVERTERS AS STATE-MACHINES ================
#:::::::::::::::::::::: Subsections 4.2.1 : States and Transitions ::::::::::::::::::::::
set STREAMS_NGFeed := FeedMatte Flux Ferronickel;
set ELEMENTS := Fe Ni Co Cu S Si Ca Al Mg O N;
```

```
set SPECIES_Prod := FeS Ni3S2 CoS Cu2S Cu_Liq Fe2SiO4 Fe3O4 NiO CoO Cu2O SiO2 CaO Al2O3
    MgO O2 N2 SO2;
set STREAMS_Prod := ConverterMatte Blister Slag Offgas;
param CMatteSymbol := ConverterMatte;
param BlisterSymbol := Blister;
param SlagSymbol := Slag;
param OffgasSymbol := Offgas;

set SPECIES[ConverterMatte] := FeS Ni3S2 CoS Cu2S;
set SPECIES[Blister] := Cu_Liq;
set SPECIES[Slag] := Fe2SiO4 Fe3O4 NiO CoO Cu2O SiO2 CaO Al2O3 MgO;
set SPECIES[Offgas] := O2 N2 SO2;

set FLOWS_NG := FeedMatteFlow FluxCharge FerronickelFlow FluxBlow ProductMatteFullLadle
    FerroslagFullLadle ProductMattePartialLadle FerroslagPartialLadle;

set FLOWS_Ch := FeedMatteFlow FluxCharge FerronickelFlow;
set FLOWS_NGBlow := FluxBlow;

set FLOWS[FeedMatte] := FeedMatteFlow;
set FLOWS[Flux] := FluxCharge FluxBlow;
set FLOWS[Ferronickel] := FerronickelFlow;
set FLOWS[ConverterMatte] := ProductMatteFullLadle ProductMattePartialLadle;
set FLOWS[Slag] := FerroslagFullLadle FerroslagPartialLadle;

set FLOWS_M_Ch := FeedMatteFlow;
set FLOWS_SM_Ch := ;
set FLOWS_M_DCh := ProductMatteFullLadle FerroslagFullLadle;
set FLOWS_SM_DCh := ProductMattePartialLadle FerroslagPartialLadle;

param v_PSC_max := 1 160 ;

param rho :=        #in T/m
    FeedMatte       5.24623341623938
    Flux            2.7620882235714
    Ferronickel     8.2
    FeS             5.27
    Ni3S2           5.17
    CoS             5.45
    Cu2S            5.28
    Cu_Liq          7.92
    Fe2SiO4         2.5
    Fe3O4           5.2
    NiO             7.45
    CoO             5.68
    Cu2O            6
    SiO2            2.65
    CaO             3.32
    Al2O3           3.99
    MgO             3.65    ;

param m_max :=
    FeedMatte       843.40003613556
    Flux            441.934115771424
```

```
    Ferronickel     1312
    Fe              602.02688
    Ni              936.6736
    Co              714.753024
    Cu              1267.2
    S               307.5572
    Si              198.19032
    Ca              379.643328
    Al              337.8732
    Mg              352.17536
    O               608.892944823589
    N               404.939259111133
    FeS             843.2
    Ni3S2           827.2
    CoS             872
    Cu2S            844.8
    Cu_Liq          1267.2
    Fe2SiO4         400
    Fe3O4           832
    NiO             1192
    CoO             908.8
    Cu2O            960
    SiO2            424
    CaO             531.2
    Al2O3           638.4
    MgO             584
    O2              15.4183072411795
    N2              404.939259111133
    SO2             586.505641031492
    ConverterMatte  872
    Blister         1267.2
    Slag            1192    ;

param h_min := 1    -10456914.5421308   ;
param h_max := 1    1026484.15688792    ;

param u_max :=
    FeedMatteFlow               4
    ProductMatteFullLadle       7
    FerroslagFullLadle          7
    ProductMattePartialLadle    1
    FerroslagPartialLadle       1   ;

param d_IntType_min :=
    4 SlagBlow  0.5
    4 SlagBlowAndSkimWithoutAnyMoreFeedMatte    0.5
    4 ExtendProductionCycleWithoutAnyMoreFeedMatte  0.5;

param d_IntType_max (tr) :                          0   1   2   3   4   5   6   7   :=
    EndPreviousCycle                                0   0   0   0   0   0   0   0
    InitialCharge                                   0   0   12  0   0   0   0   0
    SlagBlow                                        0   0   0   0   12  0   0   0
    Skim                                            0   0   0   0   0   0   12  0
    Recharge                                        0   0   12  0   0   0   0   0
```

```
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          0   0   0   0   12  0   12  0
    ExtendProductionCycleWithoutAnyMoreFeedMatte    0   0   12  0   12  0   12  0
    EndCurrentCycle                                 0   0   0   0   0   0   12  0   ;

param v_Type_max (tr): FeedMatteFlow FluxCharge FerronickelFlow FluxBlow
        ProductMatteFullLadle FerroslagFullLadle ProductMattePartialLadle
        FerroslagPartialLadle :=
    EndPreviousCycle                                0 0 0 0 0 0 0 0
    InitialCharge                                   160 160 4.878 0 0 0 0 0
    SlagBlow                                        0 0 0 160 0 0 0 0
    Skim                                            0 0 0 0 0 160 0 160
    Recharge                                        160 160 160 0 0 0 0 0
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          0 0 0 160 0 160 0 160
    ExtendProductionCycleWithoutAnyMoreFeedMatte    0 160 160 160 0 160 0 160
    EndCurrentCycle                                 0 0 0 0 160 0 160 0 ;

param u_Type_max (tr): FeedMatteFlow ProductMatteFullLadle FerroslagFullLadle
        ProductMattePartialLadle FerroslagPartialLadle :=
    EndPreviousCycle                                0 0 0 0 0
    InitialCharge                                   4 0 0 0 0
    SlagBlow                                        0 0 0 0 0
    Skim                                            0 0 7 0 1
    Recharge                                        1 0 0 0 0
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          0 0 7 0 1
    ExtendProductionCycleWithoutAnyMoreFeedMatte    0 0 7 0 1
    EndCurrentCycle                                 0 7 0 1 0 ;

param v_u :=
    FeedMatteFlow           20
    ProductMatteFullLadle   20
    FerroslagFullLadle      20  ;

param v_u_max :=
    ProductMattePartialLadle    20
    FerroslagPartialLadle       20  ;


#:::::::::::::::::::::::: Subsection 4.2.2 : Converting Actions ::::::::::::::::::::::::
param d_Ch_Type :=
    InitialCharge                                   0.5
    Recharge                                        0.125
    ExtendProductionCycleWithoutAnyMoreFeedMatte    0.125   ;

param d_DCh_Type :=
    Skim                                            0.125
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          0.125
    ExtendProductionCycleWithoutAnyMoreFeedMatte    0.125
    EndCurrentCycle                                 0.5     ;


#======================== SECTION 4.3 : INTERMEDIATE COMPUTATIONS ========================
#:::::::::::::::::::: Subsections 4.3.1 : Intermediate Variables ::::::::::::::::::::::::
param m_Blast_max :=    #in T
    0   308.366144823589
```

```
    N   404.939259111133    ;

param h_Blast_min := 0 ; #in MJ
param h_Blast_max := 16758.0668631928 ;

param h_Offgas_min := -2194047.08109494 ; #in MJ
param h_Offgas_max := 525084.410598306 ;

param h_DCh_min := -10456914.5421308 ; #in MJ
param h_DCh_max := 1026484.15688792 ;

param h_Env_max :=  #in MJ
    1   1296000
    2   1296000
    3   1296000
    4   1296000
    5   1296000
    6   1296000
    7   1296000 ;

param w (tr) : Fe Ni Co Cu S Si Ca Al Mg O N :=
    FeedMatte   0.3726 0.2 0.01 0.1 0.3174 0 0 0 0 0 0
    Flux        0 0 0 0 0 0.3973 0.03573 0.02646 0.03015 0.5103 0
    Ferronickel 0.3 0.7 0 0 0 0 0 0 0 0 0
    FeS         0.6352 0 0 0 0.3648 0 0 0 0 0 0
    Ni3S2       0 0.7330 0 0 0.2670 0 0 0 0 0 0
    CoS         0 0 0.6476 0 0.3524 0 0 0 0 0 0
    Cu2S        0 0 0 0.7985 0.2015 0 0 0 0 0 0
    Cu_Liq      0 0 0 1 0 0 0 0 0 0 0
    Fe2SiO4     0.5481 0 0 0 0 0.1378 0 0 0 0.3141 0
    Fe3O4       0.7236 0 0 0 0 0 0 0 0 0.2764 0
    NiO         0 0.7858 0 0 0 0 0 0 0 0.2142 0
    CoO         0 0 0.7865 0 0 0 0 0 0 0.2135 0
    Cu2O        0 0 0 0.8882 0 0 0 0 0 0.1118 0
    SiO2        0 0 0 0 0 0.4674 0 0 0 0.5326 0
    CaO         0 0 0 0 0 0 0.7147 0 0 0.2853 0
    Al2O3       0 0 0 0 0 0 0 0.5292 0 0.4708 0
    MgO         0 0 0 0 0 0 0 0 0 0.6030 0.3970 0
    O2          0 0 0 0 0 0 0 0 0 0 1 0
    N2          0 0 0 0 0 0 0 0 0 0 0 1
    SO2         0 0 0 0 0.5005 0 0 0 0 0.4995 0 ;

param w_Feed_H :=       #in MJ/T
    FeedMatteFlow       358.213067563604    # evaluated at matte feed temperature
    FluxCharge          -14169.9586946575   # evaluated at cold charge temperature
    FerronickelFlow     -123.8677           # evaluated at cold charge temperature
    FluxBlow            -14169.9586946575   ; # evaluated at cold charge temperature


#:::::::::::::::::::::: Subsection 4.3.2 : Blast Elemental Masses ::::::::::::::::::::::
param m_Blast_dot :=                                #in T/h
    O SlagBlow                                      25.6968
    N SlagBlow                                      33.7446
    O SlagBlowAndSkimWithoutAnyMoreFeedMatte        25.6968
```

```
    N SlagBlowAndSkimWithoutAnyMoreFeedMatte        33.7446
    O ExtendProductionCycleWithoutAnyMoreFeedMatte  25.6968
    N ExtendProductionCycleWithoutAnyMoreFeedMatte  33.7446 ;


#:::::::::::::::::::::: Subsection 4.3.3 : Product Species Masses ::::::::::::::::::::::
set REGIMES := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime CopperBlowRegime
    CopperOverblowRegime;
set SPECIES_RgProd := FeS Ni3S2 CoS Cu2S Cu_Liq NiO CoO Cu2O;

set REGIMES_spec[FeS] := SlagBlowRegime;
set REGIMES_spec[Ni3S2] := SlagBlowRegime NickelOverblowRegime;
set REGIMES_spec[CoS] := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime;
set REGIMES_spec[Cu2S] := SlagBlowRegime NickelOverblowRegime CobaltOverblowRegime
    CopperBlowRegime;
set REGIMES_spec[Cu_Liq] := CopperBlowRegime CopperOverblowRegime;
set REGIMES_spec[NiO] := NickelOverblowRegime CobaltOverblowRegime CopperBlowRegime
    CopperOverblowRegime;
set REGIMES_spec[CoO] := CobaltOverblowRegime CopperBlowRegime CopperOverblowRegime;
set REGIMES_spec[Cu2O] := CopperOverblowRegime;

set TYPES_PSC_Prod :=  SlagBlow Skim Recharge SlagBlowAndSkimWithoutAnyMoreFeedMatte
            ExtendProductionCycleWithoutAnyMoreFeedMatte EndCurrentCycle;

param OXYGEN_EFFICIENCY := 0.95 ;
param FERROSLAG_RATIO := 2 ;

param OSymbol := O;
param O2Symbol := O2;
param Fe2SiO4Symbol := Fe2SiO4;
param Fe3O4Symbol := Fe3O4;


#:::::::::::::::::::::::::: Subsection 4.3.4 : Blast Heat ::::::::::::::::::::::::::
param h_Blast_dot :=                                #in MJ/h
    SlagBlow                                        1396.50557193274
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          1396.50557193274
    ExtendProductionCycleWithoutAnyMoreFeedMatte    1396.50557193274    ;


#:::::::::::::::::::::::::: Subsection 4.3.5 : Offgas Heat ::::::::::::::::::::::::::
param w_Offgas_H (tr) :                             O2          N2          SO2 :=
    EndPreviousCycle                                0           0           0
    InitialCharge                                   0           0           0
    SlagBlow                                        1240.56     1296.70     -3740.88
    Skim                                            0           0           0
    Recharge                                        0           0           0
    SlagBlowAndSkimWithoutAnyMoreFeedMatte          1240.56     1296.70     -3740.88
    ExtendProductionCycleWithoutAnyMoreFeedMatte    1240.56     1296.70     -3740.88
    EndCurrentCycle                                 0           0           0       ;


#:::::::::::::::::::::::: Subsection 4.3.6 : Discharge Heat ::::::::::::::::::::::::
param w_DCh_H : Skim SlagBlowAndSkimWithoutAnyMoreFeedMatte
```

```
        ExtendProductionCycleWithoutAnyMoreFeedMatte :=
    FeedMatte       379.740559318401    379.740559318401    379.740559318401
    Flux            -12669.6798817374   -12669.6798817374   -12669.6798817374
    Ferronickel     438.9323            438.9323            438.9323
    FeS             122.21539074053     122.21539074053     122.21539074053
    Ni3S2           961.419731551783    961.419731551783    961.419731551783
    CoS             927.271139828653    927.271139828653    927.271139828653
    Cu2S            250.984342504571    250.984342504571    250.984342504571
    Cu_Liq          800.160600195134    800.160600195134    800.160600195134
    Fe2SiO4         -5733.28422642635   -5733.28422642635   -5733.28422642635
    Fe3O4           -3606.67074176221   -3606.67074176221   -3606.67074176221
    NiO             -2378.91470463181   -2378.91470463181   -2378.91470463181
    CoO             -2182.36969709799   -2182.36969709799   -2182.36969709799
    Cu2O            -551.994700502616   -551.994700502616   -551.994700502616
    SiO2            -12640.4612884166   -12640.4612884166   -12640.4612884166
    CaO             -10053.5226944017   -10053.5226944017   -10053.5226944017
    Al2O3           -14949.489977358    -14949.489977358    -14949.489977358
    MgO             -13502.7430599066   -13502.7430599066   -13502.7430599066   ;


#:::::::::::::::::::: Subsection 4.3.7 : Environmental Heat Losses ::::::::::::::::::::
param h_EnvType_dot (tr): 0 1 2 3 4 5 6 7 := #in MJ/h
    EndPreviousCycle    0 0 0 0 0 0 0 0
    InitialCharge       0 0 0 0 0 0 0 0
    SlagBlow            108000 108000 108000 108000 108000 108000 108000 108000
    Skim                108000 108000 108000 108000 108000 108000 108000 108000
    Recharge            108000 108000 108000 108000 108000 108000 108000 108000
    SlagBlowAndSkimWithoutAnyMoreFeedMatte  108000 108000 108000 108000 108000 108000
        108000 108000
    ExtendProductionCycleWithoutAnyMoreFeedMatte 108000 108000 108000 108000 108000
        108000 108000 108000
    EndCurrentCycle     0 0 0 0 0 0 0 0 ;


#======================== SECTION 4.4 : FORWARD COMPUTATIONS ==========================
#:::::::::::::::::::: Subsection 4.4.1 : Retained Feed Masses ::::::::::::::::::::::::
#::::::::::::::::::: Subsection 4.4.2 : Retained Product Species Masses ::::::::::::::::
#::::::::::::::::::: Subsection 4.4.3 : Forward Heat Computation ::::::::::::::::::::::

#==================== SECTION 4.5 : FEASIBLE CONVERTER TRANSITIONS =====================
#::::::::: Subsection 4.5.1 : Direct Transition Constraints in General Linear Form ::::::

#::::::::::::::::::::: Subsection 4.5.2 : Bath Composition Constraints ::::::::::::::::::
param w_strm_O_max :=
    #Restricted use of excess flux
        SiO2 Slag Skim 0.1 ;

param w_strm_Blow_max :=
    #Restricted use of excess flux
        SiO2 Slag SlagBlowAndSkimWithoutAnyMoreFeedMatte        0.1
        SiO2 Slag ExtendProductionCycleWithoutAnyMoreFeedMatte  0.1
    #No Overblow
        NiO Slag SlagBlow 0
        NiO Slag SlagBlowAndSkimWithoutAnyMoreFeedMatte 0
```

```
        NiO Slag ExtendProductionCycleWithoutAnyMoreFeedMatte 0;


#:::::::::::::::::::::::: Subsection 4.5.3 : Volume Constraints ::::::::::::::::::::::::

#:::::::::::::::::::: Subsection 4.5.4 : Temperature Constraints ::::::::::::::::::::::
param w_Blow_H_min : EndPreviousCycle InitialCharge SlagBlow Skim Recharge
        SlagBlowAndSkimWithoutAnyMoreFeedMatte
        ExtendProductionCycleWithoutAnyMoreFeedMatte EndCurrentCycle :=
    FeedMatte       250.58 250.58 250.58 250.58 250.58 250.58 250.58 250.58
    Flux            -12951 -12951 -12951 -12951 -12951 -12951 -12951 -12951
    Ferronickel     354.51 354.51 354.51 354.51 354.51 354.51 354.51 354.51
    FeS             -5.8608 -5.8608 -5.8608 -5.8608 -5.8608 -5.8608 -5.8608 -5.8608
    Ni3S2           817.70 817.70 817.70 817.70 817.70 817.70 817.70 817.70
    CoS             788.81 788.81 788.81 788.81 788.81 788.81 788.81 788.81
    Cu2S            149.58 149.58 149.58 149.58 149.58 149.58 149.58 149.58
    Cu_Liq          711.24 711.24 711.24 711.242 711.24 711.24 711.24 711.24
    Fe2SiO4         -5945.8 -5945.8 -5945.8 -5945.8 -5945.8 -5945.8 -5945.8 -5945.8
    Fe3O4           -3825.6 -3825.6 -3825.6 -3825.6 -3825.6 -3825.6 -3825.6 -3825.6
    NiO             -2523.5 -2523.5 -2523.5 -2523.5 -2523.5 -2523.5 -2523.5 -2523.5
    CoO             -2339.1 -2339.1 -2339.1 -2339.1 -2339.1 -2339.1 -2339.1 -2339.1
    Cu2O            -672.81 -672.81 -672.81 -672.81 -672.81 -672.81 -672.81 -672.81
    SiO2            -12930 -12930 -12930 -12930 -12930 -12930 -12930 -12930
    CaO             -10279 -10279 -10279 -10279 -10279 -10279 -10279 -10279
    Al2O3           -15204 -15204 -15204 -15204 -15204 -15204 -15204 -15204
    MgO             -13735 -13735 -13735 -13735 -13735 -13735 -13735 -13735 ;

param w_Blow_H_max : EndPreviousCycle InitialCharge SlagBlow Skim Recharge
        SlagBlowAndSkimWithoutAnyMoreFeedMatte
        ExtendProductionCycleWithoutAnyMoreFeedMatte EndCurrentCycle :=
    FeedMatte       394.09 394.09 394.09 394.09 394.09 394.09 394.09 394.09
    Flux            -12637 -12637 -12637 -12637 -12637 -12637 -12637 -12637
    Ferronickel     448.31 448.31 448.31 448.31 448.31 448.31 448.31 448.31
    FeS             136.45 136.45 136.45 136.45 136.45 136.45 136.45 136.45
    Ni3S2           977.39 977.39 977.39 977.39 977.39 977.39 977.39 977.39
    CoS             942.66 942.66 942.66 942.66 942.66 942.66 942.66 942.66
    Cu2S            262.25 262.25 262.25 262.25 262.25 262.25 262.25 262.25
    Cu_Liq          810.04 810.04 810.04 810.04 810.04 810.04 810.04 810.04
    Fe2SiO4         -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7 -5709.7
    Fe3O4           -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6 -3581.6
    NiO             -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6 -2362.6
    CoO             -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6 -2164.6
    Cu2O            -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24 -538.24
    SiO2            -12607 -12607 -12607 -12607 -12607 -12607 -12607 -12607
    CaO             -10028 -10028 -10028 -10028 -10028 -10028 -10028 -10028
    Al2O3           -14920 -14920 -14920 -14920 -14920 -14920 -14920 -14920
    MgO             -13477 -13477 -13477 -13477 -13477 -13477 -13477 -13477 ;

param w_DCh_H_min : EndCurrentCycle :=  #in MJ/T
    FeedMatte       322.333914638943
    Flux            -12797.0498513263
    Ferronickel     401.4123
    FeS             65.2926288249346
    Ni3S2           897.544009371792
```

```
    CoS             865.731457640451
    Cu2S            205.915385437021
    Cu_Liq          760.638364334498
    Fe2SiO4         -5827.73437710866
    Fe3O4           -3705.37822518764
    NiO             -2443.67823153721
    CoO             -2252.71768244752
    Cu2O            -606.357787312864
    SiO2            -12771.4558084642
    CaO             -10155.2236001187
    Al2O3           -15064.3120428775
    MgO             -13606.7126396395 ;

param w_DCh_H_max : EndCurrentCycle :=  #in MJ/T
    FeedMatte       394.092220488266
    Flux            -12637.2684712361
    Ferronickel     448.3123
    FeS             136.446081219429
    Ni3S2           977.38866209678
    CoS             942.656060375703
    Cu2S            262.251581771458
    Cu_Liq          810.041159160293
    Fe2SiO4         -5709.67168875578
    Fe3O4           -3581.64276106801
    NiO             -2362.60333645964
    CoO             -2164.59758386118
    Cu2O            -538.237252116479
    SiO2            -12607.0954189282
    CaO             -10027.7312463732
    Al2O3           -14920.403009648
    MgO             -13476.6130469213   ;


#:::::: Subsection 4.5.5 : Indirect Transition Constraints in General Linear Form :::::::

#============ SECTION 4.6 : OPTIMIZATION OBJECTIVES AND GLOBAL CONSTRAINTS ==============
#:::::: Subsection 4.6.1 : Optimization of Nongaseous Flows and of Transition Types :::::
param c_v :=    FerronickelFlow 1;


#:::::::: Subsection 4.6.2 : Limiting of Nongaseous Flows and of Transition Types :::::::
set GLOBAL_CONSTRAINTS_PSC := MinimumFeedMatteFlow MaximumRecharges;

param a_global_v :=
    #Place lower limit on feed matte
        MinimumFeedMatteFlow FeedMatteFlow -1;

param a_global_BType :=
    #Limit the number of recharges
        MaximumRecharges Recharge 1;

param b_global :=
    #Limit the number of recharges
        MinimumFeedMatteFlow    -120
```

```
    #Limit the number of recharges
        MaximumRecharges        2   ;


#====================== SECTION 5.1 : THE SINGLE-CYCLE PSC PROBLEM ======================
#:::::::::: Subsection 5.1.1 : MILP Formulation of the Single-Cycle PSC Problem ::::::::
param EndPreviousCycleSymbol := EndPreviousCycle;
param EndCurrentCycleSymbol := EndCurrentCycle;


#:::::::::::::: Subsection 5.1.2 : Critical Overlap Decomposition of a Cycle ::::::::::::
param n_PSC_system := 2 ;
param n_Crit_max := 1 ;

set TYPES_PSC_PreCrit := ;
set TYPES_PSC_Crit := InitialCharge SlagBlow Skim Recharge;


#:::::::::::: Subsection 5.1.3 : Critical Overlap Decomposition of a Cycle ::::::::::::::

#:::::::::: Subsection 5.1.4 : Maximizing the Productivity of a Single Cycle ::::::::::::


#=============================================================================================
#=============================================================================================
#=============================================================================================
```

C.3 *run* Files

The following script corresponds to the *run* file that was used in all of the computations presented in Chapter 5. The first section of the script, "User Options", allows the user to select the type of computation by making small modifications to the text.

There are three different computation modes. Firstly, the "RatioObjective" mode is to maximize the ratio objective described by Equation 5.18, by testing 52 different $\overline{d}_{\text{Crit}}$ values. Secondly, the "TimeTrials" mode repeats the work of the "RatioObjective" mode numerous times, in order to obtain time data; the number of trials can be set by the user. The third mode, "Fix_dCritMax", isolates a single value of $\overline{d}_{\text{Crit}}$, which can be set by the user.

```
#=============================================================================================
#=============================================================================================
#=============================================================================================

reset;
#------------------------------------ User Options ------------------------------------
param UserOption_ComputationModeSelect symbolic;
param UserOption_Fix_dCritMax;
param UserOption_NumberOfTimeTrials;
```

```
#(1) Select computation mode ("RatioObjective", "TimeTrials", or "Fix_dCritMax"):
let UserOption_ComputationModeSelect := "RatioObjective";

#(2) Enter the fixed value of d_Crit_max (relevant for the "Fix_dCritMax" mode only):
let UserOption_Fix_dCritMax := 12;

#(3) Enter the number of time trials (relevant for the "TimeTrials" mode only):
let UserOption_NumberOfTimeTrials := 20;


#------------------------------------ First Printout ----------------------------------
printf "\n*************************************************************************\n";
printf "*************************************************************************\n";
printf "*************************************************************************\n";
printf "*************** Single-Cycle Peirce-Smith Converter Problem *************\n";
printf "*************************************************************************\n";
printf "*************************************************************************\n";
printf "*************************************************************************\n\n";

printf "\n\nInput problem data will be read from PSCP.dat\n";
printf "Output solution data will be written to PSCP.csv\n\n";

if UserOption_ComputationModeSelect = "Fix_dCritMax" then
    printf "d_Crit_max has been fixed to %5.2f\n\n", UserOption_Fix_dCritMax;

if UserOption_ComputationModeSelect = "TimeTrials" then
    printf "The optimization will be repeated %d times.\n\n",
    UserOption_NumberOfTimeTrials;


#------------------------------------ AMPL Options ------------------------------------
#Suppress exiting due to warnings
option eexit -100;

#Display solver messages and statistics
option solver_msg 1;
option show_stats 1;

#Select CPLEX as the solver
option solver cplex;

#Do not transmit results of one solution to for initial solution of next solution
option send_statuses 0;


#----------------------- Load Model, Load Data and Declare Parameters -----------------
#Load model and data
model PSCP.mod;
data PSCP.dat;

#Declare "parameters" that are used in the time trials and iterations
param ComputationTime {1..2, 1..UserOption_NumberOfTimeTrials} default 0;
    #Row 1 is main computation time, Row 2 is ancillary computation time
```

```
param BestIteration;
param BestRatioObjectiveValue;
param d_Crit_max {i in 1..52} default (i/4);
param NumeratorObjectiveValue {1..52} default 0;
param DenominatorObjectiveValue {1..52} default 0;
param RatioObjectiveValue {1..52} default 0;


#------------------------------ Deactivate Certain Constraints -------------------------
#Deactivate dependency constraints
drop Chp4_Eq008_DependencyClause;
drop Chp4_Eq009_TypeSupportConsistency;
drop Chp4_Eq010_SupportNoMoreThanOneAssignment;
drop Chp4_Eq011_SimultaneousDuration ;
drop Chp4_Eq012_SimultaneousDuration;
drop Chp4_Eq013_SimultaneousCompletionTime;
drop Chp4_Eq014_SimultaneousCompletionTime;

#Deactivate critical dominance condition
drop Chp5_Eq12_CriticalDominance;


#---------------------- Apply Time Trials and Iterations (If Necessary) -----------------
#Adjust the number of time trials to 1 unless in "TimeTrials" mode
if UserOption_ComputationModeSelect <> "TimeTrials" then
    let UserOption_NumberOfTimeTrials := 1;

#Apply time trials and iterations, depending on computation mode.
if UserOption_ComputationModeSelect <> "Fix_dCritMax" then{
    #Time Trial Loop
    for {Trial in 1..UserOption_NumberOfTimeTrials} {
        let BestIteration := 1;
        let BestRatioObjectiveValue := 0;
        let ComputationTime [1,Trial] := time();

        #Perform first sweeping (1 to 48), and second sweeping (49 to 52),
        for {Iteration in 1..52}{
            #Print Trial and Iteration Numbers
            printf "\n\n*********************************************************";
            printf "***************\n";
            if (UserOption_ComputationModeSelect = "TimeTrials") then
                printf "Trial %d, ", Trial;
            printf "Iteration %d\n", Iteration;

            #Impose upper bound on critical duration (d_Crit)
            let d_Crit_max_actual := d_Crit_max[Iteration];

            #Maximize production
            drop Chp5_Eq22_MaintainProduction;
            objective f1;
            solve;

            if f1.result = "solved" then {
            #If production was successfully maximized, then proceed to minimize
```

```
            #critical duration and record values the objective values.
                let f1_star_actual := f1.val;
                restore Chp5_Eq22_MaintainProduction;
                objective f2;
                solve;

                let NumeratorObjectiveValue[Iteration] := f1.val;
                let DenominatorObjectiveValue[Iteration] := f2.val;
            } else {
            #Otherwise, zeros are recorded.
                let NumeratorObjectiveValue[Iteration] := 0;
                let DenominatorObjectiveValue[Iteration] := 0;
            }

            #Record the ratio objective values
            if DenominatorObjectiveValue[Iteration] > 0 then
                let RatioObjectiveValue[Iteration] :=
                                NumeratorObjectiveValue[Iteration]/
                                DenominatorObjectiveValue[Iteration];
            else let RatioObjectiveValue[Iteration] := 0;

            #Compare current RatioObjectiveValue to BestOjectiveRatioValue
            if RatioObjectiveValue[Iteration] > BestRatioObjectiveValue then{
                let BestIteration := Iteration;
                let BestRatioObjectiveValue := NumeratorObjectiveValue[Iteration]/
                                DenominatorObjectiveValue[Iteration];
            }

            if Iteration = 48 then {
            #Determine d_Crit_Max values for second sweeping
                let d_Crit_max[49] := BestIteration/4 - 1/6;
                let d_Crit_max[50] := BestIteration/4 - 1/12;
                let d_Crit_max[51] := BestIteration/4 + 1/12;
                let d_Crit_max[52] := BestIteration/4 + 1/6;
            }
        }
        let ComputationTime [1,Trial] := time() - ComputationTime [1,Trial];
    }
}

#------------------------------ Determine Optimal Solution ----------------------------
#Print out delineator, except for Fix_dCritMax mode
if UserOption_ComputationModeSelect <> "Fix_dCritMax" then{
    printf "\n\n*************************************************************************";
    printf "****\n";
    printf "Reload Optimal Solution\n";
}

#Load d_Crit_max according to the computation mode
if UserOption_ComputationModeSelect = "Fix_dCritMax" then
    let d_Crit_max_actual := UserOption_Fix_dCritMax;
else
    let d_Crit_max_actual := d_Crit_max[BestIteration];
```

```
#Time the computation for "Fix_dCritMax"
#(The other modes will already have computation time data for the main computation)
if UserOption_ComputationModeSelect = "Fix_dCritMax" then
    let ComputationTime[1,1] := time();

#Maximize Production
drop Chp5_Eq22_MaintainProduction;
objective f1;
solve;

#Minimize Critical Duration
let f1_star_actual := f1.val;
restore Chp5_Eq22_MaintainProduction;
objective f2;
solve;

#Record main computation time (for "Fix_dCritMax" mode)
if UserOption_ComputationModeSelect = "Fix_dCritMax" then
    let ComputationTime[1,1] := time() - ComputationTime[1,1];


#---------------------------- A Posteriori Ancillary Assignment ------------------------
#Fix Gantt variables for PSC, leaving the ancillary classes unfixed
fix {k in TYPES_PSC, (l2,l3) in ASSIGNMENTS_PSC} B_Type[k,PSCSymbol,l2,l3];
fix {(l2,l3) in ASSIGNMENTS_CURRENT_PSC} d[PSCSymbol,l2,l3];
fix {(l2,l3) in ASSIGNMENTS_PSC} t[PSCSymbol,l2,l3];

#Fix all remaining PSC variables
fix m_Ret;
fix h_Ret;
fix d_Int;
fix v;
fix u;
fix m;
fix m_Prod;
fix m_Blast;
fix h_Blast;
fix h_Env;
fix m_RetProd;
fix B_Rg;

#Fix the SCPS related variables
fix d_Crit;
fix d_Cycle;
fix d_CritComponent;
fix d_CycleComponent;

#Restore dependency constraints
restore Chp4_Eq008_DependencyClause;
restore Chp4_Eq009_TypeSupportConsistency;
restore Chp4_Eq010_SupportNoMoreThanOneAssignment;
restore Chp4_Eq011_SimultaneousDuration ;
restore Chp4_Eq012_SimultaneousDuration;
restore Chp4_Eq013_SimultaneousCompletionTime;
```

```
restore Chp4_Eq014_SimultaneousCompletionTime;

#Loosen presolve infeasibility detection
option presolve_eps 1e-8;

#Set constant objective
maximize constant: 0;
objective constant;

for {Trial in 1..UserOption_NumberOfTimeTrials} {
    #Print trial information, if applicable
    printf "\n\n**************************************************************************";
    printf "****\n";
    if (UserOption_ComputationModeSelect = "TimeTrials") then
        printf "Trial %d, ", Trial;
    printf "A Posteriori Ancillary Assignment\n";

    let ComputationTime[2,Trial] := time();
    solve;
    let ComputationTime[2,Trial] := time() - ComputationTime[2,Trial];
}


#-------------------------- Print Objectives and Computation Times ----------------------
printf "\n\n**************************************************************************";
printf "\n\nf1 = %5.3f \n", f1;
printf "f2 = %5.3f \n", f2;
if f2 <> 0 then
    printf "f1/f2 = %5.3f \n\n", f1/f2;
else
    printf "f1/f2 = NaN \n\n";

printf "Main Computation Time = %d s\n", ComputationTime[1,1];
printf "Ancillary Computation Time = %d s\n\n\n", ComputationTime[2,1];


#----------------------------------- Create CSV File ---------------------------------
#Print the computation times
printf "Time Trials,\n,Trial," >PSCP.csv;
for {Trial in 1..UserOption_NumberOfTimeTrials}
    printf "%d,",Trial >PSCP.csv;
printf "\n,Main Computation Time," >PSCP.csv;
for {Trial in 1..UserOption_NumberOfTimeTrials}
    printf "%f,",ComputationTime[1,Trial] >PSCP.csv;
printf "\n,Ancillary Computation Time," >PSCP.csv;
for {Trial in 1..UserOption_NumberOfTimeTrials}
    printf "%f,",ComputationTime[2,Trial] >PSCP.csv;

#Print the optimization objectives
printf "\n\n\nOptimization Objectives,\n,Iteration," >PSCP.csv;
for {Iteration in 1..52}
    printf "%d,",Iteration >PSCP.csv;
printf "\n,d_Crit_max," >PSCP.csv;
for {Iteration in 1..52}
```

```
    printf "%f,",d_Crit_max[Iteration] >PSCP.csv;
printf "\n,f1*," >PSCP.csv;
for {Iteration in 1..52}
    printf "%f,",NumeratorObjectiveValue[Iteration] >PSCP.csv;
printf "\n,f2*," >PSCP.csv;
for {Iteration in 1..52}
    printf "%f,",DenominatorObjectiveValue[Iteration] >PSCP.csv;
printf "\n,f1*/f2*," >PSCP.csv;
for {Iteration in 1..52}
    printf "%f,",RatioObjectiveValue[Iteration] >PSCP.csv;

#Print the Ancillary Assignment Variables
for {i in CLASSES diff {PSCSymbol}}{
    printf "\n\n\n%s Assignments,\n,Assignment Number,",i >PSCP.csv;
    for {k in 1..n_Asgn_max[i]}
        printf "%d,",k >PSCP.csv;
    printf "\n,Duration (hr)," >PSCP.csv;
    for {k in 1..n_Asgn_max[i]}
        printf "%f,",d[i,1,k] >PSCP.csv;
    printf "\n,Finishing Time (hr)," >PSCP.csv;
    for {k in 1..n_Asgn_max[i]}
        printf "%f,",t[i,1,k] >PSCP.csv;
    printf "\n,Assignment Type," >PSCP.csv;
    for {kp in 1..n_Asgn_max[i]}
        for {k in TYPES[i]}
            if B_Type[k,i,1,kp] = 1 then printf "%s,",k >PSCP.csv;
}
#Print the Converter Transition Variables
printf "\n\n\nConverter Transitions,\n,Transition Number," >PSCP.csv;
for {k in 1..n_Asgn_max[PSCSymbol]}
    printf "%d,",k >PSCP.csv;
printf "\n\nGantt Structure,\n,Duration (hr)," >PSCP.csv;
for {k in 1..n_Asgn_max[PSCSymbol]}
    printf "%f,",d[PSCSymbol,1,k] >PSCP.csv;
printf "\n,Finishing Time (hr)," >PSCP.csv;
for {k in 1..n_Asgn_max[PSCSymbol]}
    printf "%f,",t[PSCSymbol,1,k] >PSCP.csv;
printf "\n,Transition Type," >PSCP.csv;
for {kp in 1..n_Asgn_max[PSCSymbol]}
    for {k in TYPES_PSC}
        if B_Type[k,PSCSymbol,1,kp] = 1 then printf "%s,",k >PSCP.csv;
printf "\n\nSegment Durations (hr)" >PSCP.csv;
for {i in 0..7}{
    printf "\n,%d,",i >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",d_Int[i,1,k] >PSCP.csv;
}
printf "\n\nCharge Delivery Volumes (m)" >PSCP.csv;
for {j in FLOWS_Ch}{
    printf "\n,%s,",j >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",v[j,1,k] >PSCP.csv;
}
printf "\n\nBlow Delivery Volumes (m)" >PSCP.csv;
```

```
for {j in FLOWS_NGBlow}{
    printf "\n,%s,",j >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",v[j,1,k] >PSCP.csv;
}
printf "\n\nDischarge Delivery Volumes (m)" >PSCP.csv;
for {j in FLOWS_DCh}{
    printf "\n,%s,",j >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",v[j,1,k] >PSCP.csv;
}
printf "\n\nCharge Delivery Units" >PSCP.csv;
for {j in FLOWS_MSM_Ch}{
    printf "\n,%s,",j >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",u[j,1,k] >PSCP.csv;
}
printf "\n\nDischarge Delivery Units" >PSCP.csv;
for {j in FLOWS_MSM_DCh}{
    printf "\n,%s,",j >PSCP.csv;
    for {k in 1..n_Asgn_max[PSCSymbol]}
        printf "%f,",u[j,1,k] >PSCP.csv;
}
printf "\n\nTransition-Type Determinants" >PSCP.csv;
for {k in TYPES_PSC}{
    printf "\n,%s,",k >PSCP.csv;
    for {kp in 1..n_Asgn_max[PSCSymbol]}
        printf "%d,",B_Type[k,PSCSymbol,1,kp] >PSCP.csv;
}
#Close csv file
close PSCP.csv;

#=============================================================================================
#=============================================================================================
#=============================================================================================
```